A stabilized multidomain partition of unity approach to solving incompressible viscous flow

Maximilian Balmus[a,*], Johan Hoffman[b], André Massing[c], David A. Nordsletten[a,d]

[a] *Department of Biomedical Engineering, School of Imaging Sciences and Biomedical Engineering, King's College London, King's Health Partners, London SE1 7EH, United Kingdom*
[b] *Division of Computational Science and Technology, KTH Royal Institute of Technology, SE-10044, Stockholm, Sweden*
[c] *Department of Mathematical Sciences, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway*
[d] *Department of Biomedical Engineering and Cardiac Surgery, University of Michigan, NCRC B20, 2800 Plymouth Rd, Ann Arbor, 4810, USA*

Abstract

In this work we propose a new stabilized approach for solving the incompressible Navier-Stokes equations on fixed overlapping grids. This new approach is based on the partition of unity finite element method, which defines the solution fields as weighted sums of local fields, supported by the different grids. Here, the discrete weak formulation of the problem is re-set in cG(1)cG(1) stabilized form, which has the dual benefit of lowering grid resolution requirements for convection dominated flows and allowing for the use of velocity and pressure discretizations which do not satisfy the inf-sup condition. Additionally, we provide an outline of our implementation within an existing distributed parallel application and identify four key options to improve the code efficiency namely: the use of cache to store mapped quadrature points and basis function gradients, the intersection volume splitting algorithm, the use of lower order quadrature schemes, and tuning the partition weight associated with the interface elements. The new method is shown to have comparable accuracy to the single mesh boundary-fitted version of the same stabilized solver based on three transient flow tests including both 2D and 3D settings, as well as low and moderate Reynolds number flow conditions. Moreover, we demonstrate how the four implementation options have a synergistic effect lowering the residual assembly time by an order of magnitude compared to a naive implementation, and showing good load balancing properties.

Keywords: Finite element methods, Fluid-structure interaction, Overlapping domains, Partition of unity, Stabilized flow

*Corresponding author
Email addresses: maximilian.balmus@kcl.ac.uk (Maximilian Balmus), jhoffman@kth.se (Johan Hoffman), andre.massing@ntnu.no (André Massing), david.nordsletten@kcl.ac.uk (David A. Nordsletten)

1. Introduction

The construction of high quality boundary-fitted meshes for flow simulations can be a challenging endeavour in the context of complex immersed bodies. When movement and/or contact are also included, such issues are compounded by the need to maintain mesh quality, which can necessitate either adapting the grid's connectivity matrix and/or local or global re-meshing.

For many practical applications, it is generally desirable to avoid such issues altogether. The most common approach to achieving this is through interface-capturing class techniques. These methods employ unfitted fluid grids which are typically (but not necessarily) kept fixed, and thus avoid mesh distortion altogether. Classic examples include the finite difference-based immersed boundary method [1], its finite element counterparts [2, 3, 4], as well as fictitious domain methods [5, 6] based on Lagrange multiplier coupling schemes. However, while non-conformity precludes some of the main issues of boundary-fitted solvers, it can also introduces a series of limitations, depending on the approach, such as the: inability to represent pressure jumps across the fluid solid interface, loss of the boundary layer element setup, artificial solid viscosity and incompressibility [7]. To address some of these limitations a number of different solutions have been proposed in the form local mesh adaptations [8], divergence-conforming discretizations [9] and XFEM enrichment [10, 11, 12, 13, 14, 15].

An alternative option, overlapping domain techniques, can be seen as hybrid of the non-conforming/immersed methods and boundary fitted ones [16, 17, 18, 19]. In its basic form the technique is based on a three grid setup, one for the solid and two for representing the fluid solution. One of the fluid grids is boundary fitted to the solid grid with which it is coupled using an interface tracking approach. The role of the boundary fitted fluid grid is to represent the solution in the vicinity of the interface, it can be refined appropriately to capture crucial flow features such as boundary layers, and it can be employed either an Eulerian or ALE reference frame. This grid is then embedded into the background fluid grid and the two are coupled by means of interface-capturing. While initial implementations focused on the goal of simplifying mesh generation by gluing together simpler fixed components [20, 21, 22], in recent years multiple avenues for ALE and FSI applications have also been explored, including the XFEM-based mixed/hybrid Lagrange introduced by Wall, Shahmiri and colleagues [7, 18] and the Nitsche-based Cut-FEM approach, see Massing et al. [17] and Schott et al [19]. In [23], we proposed an alternative technique based on the partition of unity finite element method, where global velocity and pressure solution fields are constructed as weighted sums of locally defined fields supported by low-order mixed element discretizations. This setup avoids both the calculation of additional fields (e.g. Lagrange multiplier field) and user-defined stabilization parameters.

In many real life applications, where moderate to high Reynolds number flow regimes are observed, the use of low-order finite standard Galerkin formulations is not practical due to the presence of non-physical spurious oscillations observed in the solution. To avoid such phenomena, restrictive and computational expensive spatial and temporal refinement are required. Hence, in practice this is generally avoided through alternative discretization methods such as spectral/hp [24, 25] and stabilization methods. The former represents a type of Galerkin approach which combines both spatial refinement and the use of high order polynomial functions to achieve exponential convergence rates

provided sufficient solution regularity. Alternatively, spurious oscillations can be avoided through modification of the weak form by adding specific terms, such as weighted residual terms, as in the case of Steamline Upwind Petrov-Galerkin [26, 27, 28] and Residual-based Variational Multiscale methods [29, 30, 31], or penalty terms [32, 33, 34]. An added benefit is that these approaches can also be used to circumvent the LBB stability condition and to employ equal order discretizations for the velocity and pressure. Through its construction, the PUFEM overlapping domain technique is theoretically amenable to stabilization, though this has not been verified up to now.

In this work we introduce a stabilized, fixed-grid, version of the PUFEM solver, SPUFEM, based on the cG(1)cG(1) scheme proposed by Hoffman et al. for incompressible flows [28], a variant of the widely known SUPG/PSPG scheme [26, 27]. Furthermore, we provide an outline of our parallel implementation, including efficient interface grid generation using functionalities provided by the SUPERMESH library [35], allowing for the extension of the SPUFEM solver to complex 3D flow settings. We also identify a series of implementation options and algorithms which can improve the code's efficiency, particularly in terms of the residual assembly time and load balancing: (1) the inclusion of a memory cache for storing mapped quadrature rules, (2) the use of the Sutherland-Hodgman-based intersection clipping and volume meshing algorithm to reduce the number of interface elements, (3) the use of lower order quadrature schemes and (4) the tuning of the interface element partition weight to better account for cost discrepancies between the standard FEM element contributions, on one side, and local partition of unity contributions, on the other. Using three benchmark problems, covering both 2D and 3D flows, as well as low and moderate Reynolds number flow regimes, we demonstrate SPUFEM to have comparable accuracy to the single mesh, boundary fitted version of the solver, using the same stabilization scheme. Furthermore, we prove that significant efficiency gains can be achieved through a careful choice in our implementation options.

The rest of the paper follows this structure. We start by introducing the Navier-Stokes equations in Section 2.1 and recalling the classic mixed finite element method in Section 2.2. Furthermore, Section 2.3 provides the outline for the single mesh cG(1)cG(1) method. In the follow up Section 2.4, we present the mixed element partition of unity approach for a two mesh setup and using the cG(1)cG(1) formulation, we define the new stabilized version of the PUFEM solver (SPUFEM). In Section 3 we discuss the implementation of partition of unity in general, allowing for the extension of the method to handle complex 3D cases. In Section 4, we present both 2D and 3D benchmark results, showing that the SPUFEM and analogous single mesh solver have comparable accuracy. Finally, we present our concluding remarks and future research directions in Section 5.

2. Methods

This section introduces the main concepts behind the SPUFEM approach. In Section 2.1, the relevant flow problem based on the non-conservative and incompressible form of the Navier-Stokes equation is defined. Subsequently, we briefly review the standard FEM formulation in Section 2.2 and the cG(1)cG(1) stabilized formulation (SFEM) in Section 2.3. Finally, the standard PUFEM and cG(1)cG(1)-based SPUFEM formulations are described analogously in Section 2.4.

2.1. The Navier-Stokes flow problem

Let $\Omega \in \mathbb{R}^d$, for $d \in \{2, 3\}$, represent the problem domain bounded by a surface Γ. The boundary can be split into two non-overlapping patches Γ_D and Γ_N, where $\Gamma = \Gamma_D \cup \Gamma_N$. Here, Γ_D and Γ_N represent the regions of the boundary associated with Dirichlet and Neumann boundary conditions, respectively. In anticipation of the PUFEM and SPUFEM problems, let us distinguish $\Gamma_o \subset \Gamma_D$, a portion of the boundary corresponding to the surface of a submerged obstacle, where no-slip conditions are generally applied, see Fig. 1. We also introduce a finite time interval $I = [0, T]$ over which the flow is observed. Thus, the NSE-based problem may be presented as follows: find the unknown velocity and pressure fields $(\mathbf{v}, p)$ which satisfy

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} + \rho \boldsymbol{v} \cdot \nabla \boldsymbol{v} - \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{v}, p) = \mathbf{0} \qquad \text{in } \Omega \times I, \tag{1a}$$

$$\nabla \cdot \boldsymbol{v} = 0 \qquad \text{in } \Omega \times I, \tag{1b}$$

$$\boldsymbol{v}(\cdot, 0) = \mathbf{u}_0 \qquad \text{in } \Omega, \tag{1c}$$

$$\boldsymbol{v} = \mathbf{u}_D \qquad \text{on } \Gamma_D \times I, \tag{1d}$$

$$\boldsymbol{\sigma} \cdot \hat{\mathbf{n}} = \mathbf{t}_N \qquad \text{on } \Gamma_N \times I, \tag{1e}$$

where ρ is the density parameter, $\boldsymbol{\sigma} = \mu(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T) - p\mathbf{I}_d$ denotes the Cauchy stress tensor, and μ is the dynamic viscosity. Furthermore, fields $\mathbf{u}_0$, $\mathbf{u}_D$ and $\mathbf{t}_N$ are used to impose the initial condition as well as the Dirichlet and Neumman boundary conditions, respectively.

2.2. The standard FEM formulation

In anticipation of the outline for the standard FEM formulation, we first introduce: Ω^h, see Fig. 1, a discrete representation of the domain, $\Gamma^h = \Gamma^h_D \cup \Gamma^h_N$, the boundary of Ω^h, and $\mathcal{T}^h = \{e_n\}_{n=1}^N$, the set of all non-overlapping simplex elements comprising the FEM grid. Similarly, we define a discretization of the time interval defined by $N_t + 1$ equidistant time points, denoted as $0 = t_0 < t_1 < \ldots < t_{N_t} = T$.

To satisfy the discrete LBB-stability condition, we chose to construct the finite element solution using simplex-based $\mathbb{P}^2 - \mathbb{P}^1$ Taylor-Hood elements. [36] Thus, the resulting discrete velocity and pressure function space can be written as

$$\boldsymbol{\mathcal{V}}^h = \left[S^2(\mathcal{T}^h)\right]^d \text{ and } \mathcal{W}^h = S^1(\mathcal{T}^h),$$

respectively, where $S^k(\mathcal{T}^h)$ is used to denote the function space comprised of piece-wise polynomial functions of order k supported by the element set $\mathcal{T}^h$:

$$S^k(\mathcal{T}^h) = \left\{ f : \Omega^h \to \mathbb{R} \mid f \in C^0(\Omega^h) \text{ and } f|_e \in \mathbb{P}^k(e) \ \forall e \in \mathcal{T}^h \right\}. \tag{2}$$

Incorporating Dirichlet homogeneous boundary conditions, we obtain the velocity test and trial function spaces

$$\boldsymbol{\mathcal{V}}^h_D = \left\{ \boldsymbol{v} \in \boldsymbol{\mathcal{V}}^h \mid \boldsymbol{v} = \boldsymbol{\pi}_h(\mathbf{u}_D) \text{ on } \Gamma^h_D \right\}, \tag{3}$$

$$\boldsymbol{\mathcal{V}}^h_0 = \left\{ \boldsymbol{v} \in \boldsymbol{\mathcal{V}}^h \mid \boldsymbol{v} = \mathbf{0} \text{ on } \Gamma^h_D \right\}, \tag{4}$$

where $\boldsymbol{\pi}_h(\mathbf{u}_D)$ is an appropriate discrete representation of the Dirichlet condition field.

Finally, using the implicit θ-step discretization scheme [37], the discrete weak problem for a given time step n may be written as follows: find $(\boldsymbol{v}_h^{n+1}, p_h^{n+1}) \in \boldsymbol{\mathcal{V}}_D^h \times \mathcal{W}^h$ such that, for all $(\boldsymbol{w}_h, q_q) \in \boldsymbol{\mathcal{V}}_0^h \times \mathcal{W}^h$, we satisfy:

$$\begin{aligned} R(\boldsymbol{v}_h^{n+1}; p_h^{n+1}; \boldsymbol{w}_h; q_h) := & \int_{\Omega_h} \rho \left[\frac{\boldsymbol{v}_h^{n+1} - \boldsymbol{v}_h^n}{\Delta t} + \boldsymbol{v}_h^{n+\theta} \cdot \nabla \boldsymbol{v}_h^{n+\theta} \right] \cdot \boldsymbol{w}_h \; dV \\ & + \int_{\Omega_h} \sigma(\boldsymbol{v}_h^{n+\theta}, p_h^{n+1}) : \nabla \boldsymbol{w}_h + q_h \nabla \cdot \boldsymbol{v}_h^{n+\theta} \; dV \\ & - \int_{\Gamma_{N,h}} \mathbf{t}_{N,h} \cdot \boldsymbol{w}_h \; dA = 0. \end{aligned} \tag{5}$$

Here, $\theta = 0.5$ is used to denote a fractional time step, such that $\boldsymbol{v}^{n+\theta} = \theta \boldsymbol{v}^{n+1} + (1-\theta)\boldsymbol{v}^n$.

2.3. The boundary fitted cG(1)cG(1) formulation (SFEM)

Here, we consider the case of the single mesh approach, SFEM, and base the solution discretization on equal order $\mathbb{P}^1 - \mathbb{P}^1$ elements. In accordance to this, we redefine the discrete velocity space as:

$$\tilde{\boldsymbol{\mathcal{V}}}^h = \left[S^1(\mathcal{T}^h) \right]^d.$$

Furthermore, through analogy to Eq. 3 and 4, we use $\tilde{\boldsymbol{\mathcal{V}}}_D^h$ and $\tilde{\boldsymbol{\mathcal{V}}}_0^h$ to denote the subspaces of $\tilde{\boldsymbol{\mathcal{V}}}^h$ which incorporate the inhomogeneous and homogeneous Dirichlet boundary conditions, respectively.

Using the same time discretization scheme as in Section 2.2, the discrete weak formulation of the flow problem for a given time step n becomes: find $(\boldsymbol{v}_h^{n+1}, p_h^{n+1}) \in \tilde{\boldsymbol{\mathcal{V}}}_D^h \times \mathcal{W}^h$ such that for all $(\boldsymbol{w}_h, q_h) \in \tilde{\boldsymbol{\mathcal{V}}}_0^h \times \mathcal{W}^h$:

$$\tilde{R}(\boldsymbol{v}_h^{n+1}; p_h^{n+1}; \boldsymbol{w}_h; q_h) := R(\boldsymbol{v}_h^{n+1}; p_h^{n+1}; \boldsymbol{w}_h; q_h) + SD_\delta(\boldsymbol{v}_h^{n+1}; p_h^{n+1}; \boldsymbol{w}_h; q_h) = 0, \tag{6}$$

where R denotes the non-stabilized residual operator introduced in Eq. 5, and SD_δ represents the set of stabilization terms used to augment the residual as given in [28]:

$$\begin{aligned} SD_\delta(\boldsymbol{v}_h^{n+1}; p_h^{n+1}; \boldsymbol{w}_h; q_h) = & \int_{\Omega_h} \delta_1 (\boldsymbol{v}_h^{n+\theta} \cdot \nabla \boldsymbol{v}_h^{n+\theta}) \cdot (\boldsymbol{v}_h^{n+\theta} \cdot \nabla \boldsymbol{w}_h + \nabla q_h) \; dV \\ & + \int_{\Omega_h} \delta_2 (\nabla \cdot \boldsymbol{v}_h^{n+\theta})(\nabla \cdot \boldsymbol{w}_h) \; dV \\ & + \int_{\Omega_h} \delta_3 \nabla p_h^{n+1} \cdot (\boldsymbol{v}_h^{n+\theta} \cdot \nabla \boldsymbol{w}_h + \nabla q_h) \; dV. \end{aligned} \tag{7}$$

Here, the first and last term of the equation, when added, are equivalent to the summation of the well-known SUPG and PSPG stabilization [26, 27], with the viscous term assumed to be zero due to the linear representation of the velocity field and time derivative considered to be negligible. Furthermore, the second term represents a least-square stabilization of the incompressibility constraint [38], also known as grad-div stabilization [39]. The stabilization parameters used to scale these terms are defined as:

$$\delta_1 = \frac{\rho h}{v_{max}}, \qquad \delta_2 = \rho h v_{max}, \qquad \text{and} \qquad \delta_3 = \frac{h}{v_{max}}, \tag{8}$$

where v_{max} is user-defined, representing the maximum expected velocity, and h is a discontinuous piece-wise constant function measuring the size of the local element.

2.4. The standard PUFEM and SPUFEM formulations

Standard PUFEM formulation: Following the problem setup introduced by us in [23], we now pose the Navier-Stokes flow problem in the PUFEM weak form based on inf-sup stable elements. To do so, we first represent the discrete domain Ω^h using the overlapping sub-domains, Ω_b^h and Ω_e^h, which we shall refer to as the background and embedded sub-domains. Associate with these, we also introduce the $\mathcal{T}_b^h$ and $\mathcal{T}_e^h$, the sets of non-overlapping simplex elements used to represent their respective grids. These grids are built to satisfy the following requirements: (1) $\Omega^h \subset \Omega_b^h$, such that any immersed surface or volume-occupying body is contained within Ω_b^h, and (2) $\Omega_e^h \subset \Omega^h$, enveloping the immersed obstacle, see Fig. 1. The latter domain also presents two boundary regions: one being $\Gamma_o^h \subset \Gamma_D^h$, the surface of the obstacle, and the other being Γ_{ff}^h, the non-conforming interface between the two fluid grids.

In the case of problems set in Eulerian reference frames, the PUFEM weak formulation does not differ significantly from the standard FEM one, with the important exception of the solution spaces. The global solution spaces are defined as weighted sums of local solution spaces, with the latter resembling the classic FEM examples. Thus, let $\boldsymbol{V}^h$ and W^h denote the PUFEM counterparts of $\boldsymbol{\mathcal{V}}^h$ and $\mathcal{W}^h$, respectively, and which can be written as:

$$\boldsymbol{V}^h = (1-\psi^h)\,\boldsymbol{V}_b^h + \psi^h\,\boldsymbol{V}_e^h, \tag{9}$$

$$W^h = (1-\psi^h) W_b^h + \psi^h W_e^h, \tag{10}$$

where the local spaces are built using $\mathbb{P}^2 - \mathbb{P}^1$ elements such that:

$$\boldsymbol{V}_e^h = \left[S^2(\mathcal{T}_e^h)\right]^d, \quad \boldsymbol{V}_b^h = \left[S^2(\mathcal{T}_b^h)\right]^d, \quad W_e^h = S^1(\mathcal{T}_e^h), \quad W_b^h = S^1(\mathcal{T}_b^h).$$

Here, the discrete weighting field, $\psi^h \in W_e^h$, $\psi^h : \Omega_e^h \to [0,1]$, is built *a priori* with the condition that $\psi^h = 0$ on Γ_{ff}^h. To simplify notation, we assume that both ψ^h and the other embedded fields have their support limited to Ω_e^h, but can be artificially extended in $\Omega^h \backslash \Omega_e^h$ by setting their values to null. Here, the polynomial representation of ψ^h is limited to piecewise linear in order to reduce the cost of PUFEM field integration. Furthermore, while the function can take any shape in theory, in practice this is limited to a transition from 0 to 1 within one layer of embedded elements. This has the effect of a quick transition from a solution dominated by the background component to one dominated by the embedded which in turn limits to a minimum the area of the problem domain were PUFEM integration is necessary, leading to further cuts in computational costs. We shall refer to the transition and constant areas of the embedded grid as

$$\Omega_{e,\psi}^h = \left\{\mathbf{x} \in \Omega_e^h : \ \nabla\psi^h(\mathbf{x}) \neq 0\right\} \text{ and } \Omega_{e,1}^h = \left\{\mathbf{x} \in \Omega_e^h : \ \psi(\mathbf{x}) = 1\right\},$$

respectively, see Fig. 2. Associated with each sub-domain, we have two element subsets $\mathcal{T}_{e,\psi}^h$ and $\mathcal{T}_{e,1}^h$, such that $\mathcal{T}_e^h = \mathcal{T}_{e,\psi}^h \cup \mathcal{T}_{e,1}^h$ and $\mathcal{T}_{e,\psi}^h \cap \mathcal{T}_{e,1}^h = \emptyset$. Distinguishing between the two types of embedded elements can be achieved by exploiting the fact that ψ^h is

supported by a piecewise linear field, evaluating the field at the centroid of each element and comparing this value to a threshold tolerance.

To guarantee solution uniqueness, it is crucial to recognise that some of the background grid's degrees of freedom do not impact the fluid solution on Ω^h and hence can take any value. For that to be the case, it is sufficient to show that the DOF's associated shape function (ϕ) satisfies the following:

$$\text{supp}(\phi) \subseteq \Omega^h_{e,1} \cup (\Omega^h_b \backslash \Omega^h), \tag{11}$$

i.e. the support of the shape function is found completely within the area covered by the solid or the area where the weight function is zero. Hence, to avoid this ambiguity, such degrees of freedom are deactivated, i.e. by being set to equal zero. In addition, for the specific case of nodal basis functions, if a background element is entirely covered by the overlapping domain, Ω^h_e, then all its DOFs will be fixed such that:

$$(\mathbf{v}^{n+1}_{b,h} - \mathbf{v}^{n+1}_{e,h})|_{\mathbf{x}} = 0 \text{ for } \mathbf{v}^{n+1}_{b,h} \in \boldsymbol{V}^h_b \text{ and } \mathbf{v}^{n+1}_{e,h} \in \boldsymbol{V}^h_e, \tag{12a}$$

$$(p^{n+1}_{b,h} - p^{n+1}_{e,h})|_{\mathbf{x}} = 0 \text{ for } p^{n+1}_{b,h} \in W^h_b \text{ and } p^{n+1}_{e,h} \in W^h_e, \tag{12b}$$

where $\mathbf{x}$ denotes the nodal coordinate, $\mathbf{v}^{n+1}_{b,h}$ and $p^{n+1}_{b,h}$ represent the background components of the velocity and pressure fields, and $\mathbf{v}^{n+1}_{e,h}$ and $p^{n+1}_{e,h}$ denote the corresponding embedded components. A more detailed presentation of the method for selection for deactivated and fixed nodes is presented in Section 3. Similarly to the single mesh formulation, $\boldsymbol{V}^h_D$ and $\boldsymbol{V}^h_0$ are used to denote sub-spaces of $\boldsymbol{V}^h$ which incorporate inhomogeneous and homogeneous Dirichlet boundary conditions, respectively. Consequently, the standard PUFEM formulation of the problem may be expressed as follows: find $(\mathbf{v}^{n+1}_h, \mathrm{p}^{n+1}_h) \in \boldsymbol{V}^h_D \times W^h_0$ such that for all $(\mathbf{w}_h, \mathrm{q}_h) \in \boldsymbol{V}^h_0 \times W^h_0$ we satisfy

$$R(\mathbf{v}^{n+1}_h; \mathrm{p}^{n+1}_h; \mathbf{w}_h; \mathrm{q}_h) = 0, \tag{13}$$

subject to node fixing and node deactivation as described above. Here, R is the same NSE residual operator used in Eq. 5.

Stabilized formulation: In order to arrive at the stabilized PUFEM (SPUFEM) formulation, we adapt the cG(1)cG(1) scheme reviewed in Section 2.3 to the partition of unity framework. To do so, we start by redefining the velocity functions space using a piecewise linear representation:

$$\tilde{\boldsymbol{V}}^h = (1 - \psi^h)\, \tilde{\boldsymbol{V}}^h_b + \psi^h\, \tilde{\boldsymbol{V}}^h_e,$$

where $\tilde{\boldsymbol{V}}^h_e = \left[S^1(\mathcal{T}^h_e)\right]$ and $\tilde{\boldsymbol{V}}^h_b = \left[S^1(\mathcal{T}^h_b)\right]$. We also extend the concept of node deactivation (based on the criterion defined in Eq. 11) and node fixing given as:

$$(\mathbf{v}^{n+1}_{b,h} - \mathbf{v}^{n+1}_{e,h})|_{\mathbf{x}} = 0 \text{ for } \mathbf{v}^{n+1}_{b,h} \in \tilde{\boldsymbol{V}}^h_b \text{ and } \mathbf{v}^{n+1}_{e,h} \in \tilde{\boldsymbol{V}}^h_e, \tag{14}$$

for $\mathbf{x}$ corresponding to the set of nodes belonging to background elements which are fully overlapped and which also intersect with $\Omega^h_{e,\psi}$. Furthermore, using the definition of $\tilde{\boldsymbol{V}}^h$, corresponding spaces incorporating the Dirichlet and homogeneous boundary are introduced and are referred to as $\tilde{\boldsymbol{V}}^h_D$ and $\tilde{\boldsymbol{V}}^h_0$, respectively.

As the original cG(1)cG(1) stabilization parameters, see Eq. 8, are dependent on the grid size, a heuristic approach is taken in order to replicate this process in the two-grid system introduced by PUFEM. Thus, considering $h_b : \Omega_b^h \to \mathbb{R}_+$ and $h_e : \Omega_e^h \to \mathbb{R}_+$, two discontinuous piecewise constant fields equal to the size of the local element, we introduce a unified element measure, $\hat{h} : \Omega_b^h \to R_+$, defined as follows:

$$\hat{h} = \begin{cases} h_b & \text{on } \Omega_b^h \backslash \Omega_e^h \\ h_e & \text{on } \Omega_{e,1}^h \\ \max(h_e, h_b) & \text{on } \Omega_{e,\psi}^h. \end{cases} \tag{15}$$

Subsequently, the SPUFEM parameters are re-evaluated based on Eq. 8, using $\hat{h}$ instead of h, and denoted as $\hat{\delta}$ to distinguish them from the SFEM case. The argument behind this definition is to allow the SPUFEM weak form to revert to the classic SFEM formulation outside of $\Omega_{e,\psi}^h$.

Having redefined the velocity space and the element size function, the SPUFEM weak form problem reads: find $(\mathbf{v}_h^{n+1}, \mathrm{p}_h^{n+1}) \in \tilde{\boldsymbol{V}}_D^h \times W^h$ such that

$$R(\mathbf{v}_h^{n+1}; \mathrm{p}^{n+1_h}; \mathbf{w}_h; \mathrm{q}_h) + SD_{\hat{\delta}}(\mathbf{v}_h^{n+1}; \mathrm{p}_h^{n+1}; \mathbf{w}_h; \mathrm{q}_h) = 0 \; \forall (\mathbf{w}, \mathrm{q}) \in \tilde{\boldsymbol{V}}_0^h \times W^h, \tag{16}$$

and subject to pre-defined deactivation and fixing of background grid nodes.

3. PUFEM/SPUFEM implementation

While the (S)PUFEM[1] formulation does not change the abstract weak form equation, instead modifying the solution and test function spaces, its main implementation challenge arises from the need to be able to: (a) compute weak form contributions combining fields supported by non-conforming grids, (b) compute mixed (S)FEM and (S)PUFEM contributions over background elements which intersect Γ_{ff}^h (which we shall refer to as cut background elements) and (c) identify the sets of background nodes which are needed to be fixed and the corresponding embedded elements required to perform this procedure. Here, by cut-background elements (or cut-elements) we mean elements which intersect the overlap area, but are not completely immersed, and hence according to our earlier definitions, have both standard and (S)PUFEM weak form contributions. Such challenges are not unique to our approach, but have been encountered and addressed in a number of other applications including XFEM, CutFEM and Galerkin projections over non-conforming grids [11, 40, 35].

In this section, we review the (S)PUFEM problem assembly procedure in general and we use this to identify the key required geometric computations. Further, we discuss the integration of the SUPERMESH library [35] functions to perform these computations efficiently and to expand the application of the method to 3D fixed grid problems.

[1] Here, (S)PUFEM is used to denote both PUFEM and SPUFEM approaches as all of the implementation topics apply to both version of the solver. However, it should be noted that all subsequent testing of their impact on run time focuses exclusively on the latter.

3.1. (S)PUFEM assembly

The general (S)PUFEM problem can be broken down into three sub-problems with distinct assembly loops: two standard (S)FEM flow sub-problems computed on portions of the embedded and background grids and a third (S)PUFEM coupling sub-problem, which requires a non-standard implementation. Here, by (S)FEM sub-problem, we refer to a region of the domain for which the assembly is by all means identical to the standard scheme, whether FEM or SFEM. To illustrate these, let us consider the task of computing the inner product of the functions $\mathbf{v}^h, \mathbf{w}^h \in \boldsymbol{V}^h$:

$$\begin{aligned}\langle \mathbf{v}^h, \mathbf{w}^h \rangle &= \int_{\Omega^h} \mathbf{v}^h \cdot \mathbf{w}^h d\Omega \\ &= \int_{\Omega^h} \left[(1-\psi^h)\mathbf{v}_b^h + \psi^h \mathbf{v}_e^h\right] \cdot \left[(1-\psi^h)\mathbf{w}_b^h + \psi^h \mathbf{w}_e^h\right] d\Omega. \end{aligned} \tag{17}$$

By using our knowledge on the behaviour of ψ^h and its relation to the problem's sub-domains, we can rewrite this operation as a sum of three terms:

$$\begin{aligned}\langle \mathbf{v}^h, \mathbf{w}^h \rangle = &\int_{\Omega^h \setminus \Omega_e^h} \mathbf{v}_b^h \cdot \mathbf{w}_b^h d\Omega + \int_{\Omega_{e,1}^h} \mathbf{v}_e^h \cdot \mathbf{w}_e^h d\Omega + \\ &+ \int_{\Omega_{e,\psi}^h} \left[(1-\psi^h)\mathbf{v}_b^h + \psi^h \mathbf{v}_e^h\right] \cdot \left[(1-\psi^h)\mathbf{w}_b^h + \psi^h \mathbf{w}_e^h\right] d\Omega. \end{aligned} \tag{18}$$

We can identify now the **first** of the standard sub-problem, namely the assembly associated with the second term of Eq. 18 which can be completed by looping over $\mathcal{T}_{e,1}^h$ and which only requires access to fields represented on the embedded grid. In contrast, the first term requires access only to the background grid but cannot be assembled using standard (S)FEM procedures due to the need to integrate over cut-elements in the background grid. The third term on the hand requires access to both background and embedded fields.

In order to explain the computations involving the background field, it is useful to split $\mathcal{T}_b^h$ into four non-intersecting sub-sets, see Fig. 4, encompassing elements which are (1) completely outside of the overlap ($\mathcal{T}_{b,b}^h$), (2) partially overlapped ($\mathcal{T}_{b,c}^h$), (3) completely overlapped but not weighted-out ($\mathcal{T}_{b,d}^h$), and (4) completely weighted out and crossing into the area of the immersed solid ($\mathcal{T}_{b,e}^h$):

$$\mathcal{T}_{b,b}^h = \left\{\tau \in \mathcal{T}_b^h \mid \tau \cap \Omega_e^h = \emptyset\right\}, \tag{19}$$

$$\mathcal{T}_{b,c}^h = \left\{\tau \in \mathcal{T}_b^h \mid \tau \cap \Omega_e^h \neq \emptyset \text{ and } \tau \not\subset \Omega_e^h\right\}, \tag{20}$$

$$\mathcal{T}_{b,d}^h = \left\{\tau \in \mathcal{T}_b^h \mid \tau \subset \Omega_e^h \text{ and } \tau \not\subset \Omega_{e,1}^h\right\}, \tag{21}$$

$$\mathcal{T}_{b,e}^h = \left\{\tau \in \mathcal{T}_b^h \mid \tau \subset (\Omega_{e,1}^h \cup \Omega_b^h \setminus \Omega^h)\right\}. \tag{22}$$

To these sets we associate sub-domains of Ω_b^h, referred to using the notation $\Omega_{b,k}^h$, for $k \in \{b, \ldots, e\}$.

Using these definitions, we first note that elements belonging $\mathcal{T}_{b,e}^h$ and $\mathcal{T}_{b,d}^h$ can be skipped from any Galerkin-type assembly procedure. This is true for the latter because all of its elements are weighted out and for the later because it coincides with the set

of elements assigned to have its degrees of freedom fixed, and hence any weak form contribution would be overwritten in the process. Furthermore, we can partially complete the process of assembling the first term in Eq. 18 by introducing the **second** standard (S)FEM sub-problem, looping over $\mathcal{T}^h_{b,d}$ and computing the weak form over $\Omega^h_{b,b}$, which is a sub-domain of $\Omega^h \backslash \Omega^h_e$. At this stage, we are left with two challenges: (1) computing the contribution over the area/volume covered by the cut background elements (outside of the overlap) and (2) computing the contributions over $\Omega^h_{e,\psi}$, neither of which can be done using standard (S)FEM procedures. A common strategy applied in literature to handle (1) is to compute the weak form directly over polygon/polyhedron resulting from subtracting the area/volume found inside the overlap from the cut-element [11, 41, 10, 40]. Instead, here we opt to first compute a standard weak form over $\mathcal{T}^h_{b,c}$, effectively adding the sub-set to the **second** (S)FEM sub-problem, and subsequently subtracting the redundant contribution from the area/volume inside the overlap. We chose this approach as it can be done at the same time as the (S)PUFEM contribution over $\Omega^h_{e,\psi}$ is added in and it avoids the need to compute contributions directly over complex, potentially non-convex sub-elements.

To address this dual challenge, we employ the SUPERMESH library to construct an interface grid, see Fig. 4. This tertiary topology itself can be split into two components depending on the computations which are required of it, either adding the (S)PUFEM contribution or subtracting the redundant cut-element contribution or both. The first of these, marked with red in Fig. 4, we denote as $\mathcal{T}^h_{i,1}$ and define as:

$$\mathcal{T}^h_{i,1} = \left\{ \tau_i \mid \exists \tau_e \in \mathcal{T}^h_{e,\psi} \text{ and } \tau_b \in \mathcal{T}^h_b,\ \tau_i \subset \tau_e \text{ and } \tau_i \subset \tau_b \right\}. \tag{23}$$

Thus, $\mathcal{T}^h_{i,1}$ is a re-triangulation of $\Omega^h_{e,\psi}$ which is conforming to both background and embedded grids. Using appropriate mapping procedure, this sub-set of the interface grid if sufficient to complete the assembly of the third term of Eq. 18. However, this set is generally insufficient to complete the process of subtracting the redundant contribution over the cut background elements. This is for example the case when there are elements in $\mathcal{T}^h_{b,c}$ which intersect with $\Omega^h_{e,1}$. To account for this, we introduce a second sub-set of the intersection grid marked with green in Fig. 4 and denoted $\mathcal{T}^h_{i,2}$:

$$\mathcal{T}^h_{i,2} = \left\{ \tau_i \mid \exists \tau_b \in \mathcal{T}^h_{b,c} \text{ and } \tau_e \in \mathcal{T}^h_{e,1},\ \tau_i \subset \tau_b \text{ and } \tau_i \subset \tau_e \right\}. \tag{24}$$

Based on these definitions, the (S)PUFEM assembly process can be summarized using the following steps:

- a standard FEM loop over $\mathcal{T}^h_{b,b} \cup \mathcal{T}^h_{b,c}$
- a second FEM loop over $\mathcal{T}^h_{e,1}$ to compute the so-called embedded problem
- a third, non-standard, loop over $\mathcal{T}^h_{i,1} \cup \mathcal{T}^h_{i,2}$ with a dual purpose:
  - compute the partition of unity inner product over $\Omega^h_{e,\psi}$ if the embedded grid parent of the intersection element is in $\mathcal{T}^h_{e,\psi}$
  - subtract the excess $\int_{\Omega^h_{b,c}} \mathbf{v}^h_b \cdot \mathbf{w}^h_b d\Omega$ if the background grid parent of the intersection element is in $\mathcal{T}^h_{b,c}$.

3.2. (S)PUFEM interface generation

The process of generating the interface grid for fixed meshes is carried out prior to the main simulation and it involves two main procedures: (1) background and embedded element collision detection and (2) the triangulation of the overlapping area for each background-embedded overlapping element pair.

Collision detection: The confirmation of the intersection detection between two simplexes by computing the intersection polygon/polyhedron is an expensive process. Furthermore, using a naive collision search approach where each background and embedded pair of elements is tested individually for intersection is impractical for large mesh sets, as it scales like $O(|\mathcal{T}_b^h| \cdot |\mathcal{T}_e^h|)$. To avoid prohibitive costs for large mesh sets, we instead make use of the quad- (2D) and oct-tree (3D) search routines provided in the SUPERMESH library [35]. These routines use the background grid to generate a hierarchical bounding box tree. Looping through the list of embedded elements, a surrogate cheaper collision test is performed by comparing the element's bounding box with those on a given level of the tree, allowing for a quick traversal. The output is a separate list for each embedded element providing the background element ID's of all potential collision candidates in the background grid, thus limiting the number of actual intersection verification tests that need to be carried out. In comparison to the naive approach, the cost of running the quad/oct-tree scales as $(|\mathcal{T}_b^h| + |\mathcal{T}_e^h|)\log(|\mathcal{T}_b^h|)$.

Intersection construction: The SUPERMESH library is also used in computing the intersection of two simplexes and the generation of local interface triangulations. The library provides two distinct approaches for carrying out these calculations: one based on the Southerland-Hodgman (SH) clipping algorithm for the 2D case and one based on Eberley's approach (EA) for the 3D case, illustrated in Fig. 3. The main distinctive characteristic is that while SH first computes the intersection and then generates a triangulation, EA performs the two tasks concomitantly. In EA, like in SH, simplex A is clipped using the faces of simplex B. However, while SH completes this process and then generates a mesh, EA triangulates the resulting polytope after each clipping, a process which is applied recursively. The main drawback of EA is that it tends to generate more intersection elements. To investigate how this might impact the 3D simulation run-time, we implemented an appropriate extension of the SH algorithm. Here, once the convex polytope is generated, the implementation uses an ad hoc triangulation based on the ear-clipping algorithm:

1. we select one node to represent the common apex of all tetrahedrons;
2. the faces opposing this node, which are also convex, are meshed using ear-clipping;
3. edges are drawn between the apex node and the face triangles to complete the triangulation process.

In our implementation both collision detection and interface construction play a role in the identification of topology subsets as described in Section 3.1. Hence, to generate $\mathcal{T}_{i,1}^h$, we loop through the elements of $T_{e,\psi}^h$ plus their lists of background element collision candidates and perform the intersection computations described above. Next, to generate $\mathcal{T}_{i,2}^h$, we loop through $T_{e,1}^h$, but limit the intersection construction to background elements which have already been shown to intersect $\Omega_{e,\psi}^h$, i.e. $\mathcal{T}_{b,c}^h \cup \mathcal{T}_{b,d}^h$. To cut grid generation costs, both $\mathcal{T}_{i,1}^h$ and $\mathcal{T}_{i,2}^h$ are discontinuous by construction, i.e. all the nodes associated with an element are unique to it. We can then distinguish between $\mathcal{T}_{b,c}^h$ and $\mathcal{T}_{b,d}^h$ by

comparing the elements' volume with the total occupied by its "children". The elements of $\mathcal{T}^h_{b,e}$ can be identified by satisfying the condition that all its vertices are found within $\Omega^h_{e,1}$. To add the background elements found within the submerged obstacle into $\mathcal{T}^h_{b,e}$, in this work we used an ad hoc process of creating a volume grid for the obstacle and temporarily adding these new elements into the embedded grid. Once the intersection grid is generated, the obstacle grid is discarded.

3.3. Sub-element weak form integration

By avoiding the need to perform integrations directly over the cut elements, (S)PUFEM computations can be performed using standard quadrature schemes, which are mapped on to the interface grids. However, the accuracy required to compute standard element contributions over $\mathcal{T}^h_{b,b}$ and $\mathcal{T}^h_{e,\psi}$ is different than that of weighted-sum contributions over $\mathcal{T}^h_{i,1}$. For comparison, a standard FEM implementation of NSE based on $\mathbb{P}^2 - \mathbb{P}^1$ elements, requires a quadrature scheme that is accurate for up to fifth order polynomials. Furthermore, an SFEM implementation based on $\mathbb{P}^1 - \mathbb{P}^1$ elements can avoid numerical integrations almost entirely by levaraging the invariance of the element mass matrix and the fact that the field gradients are piece-wise constants.

In the case of (S)PUFEM, the polynomial order of a partition of unity field is equal to that of the local field plus that of the weighting field. Assuming a linear representation of the weighting field, this raises accuracy requirements to order eight, in the case $\mathbb{P}^2 - \mathbb{P}^1$ PUFEM, and six in the case of $\mathbb{P}^1 - \mathbb{P}^1$ SPUFEM, hence further raising the cost of computing the contribution of a partition of unity (intersection) sub-element.

With the goal of reducing this effect, we will also investigate the use of sub-optimal schemes for the running of the SPUFEM implementation, i.e. schemes of accuracy order less than six. The rationale behind this is based on the following observations:

1. The background and embedded solution fields are not independent of each other, hence the second order behaviour of these functions may be much smaller in scale than the linear.
2. When integrating over a SPUFEM sub-element, all fields are only evaluated over a subsection of their parent element, meaning that a linear approximation of their local behavior may be sufficiently accurate in practice.

3.4. CHeart implementation and parallelization

The stabilized method described above has been integrated into CHeart [42], our in-house multiphysics computational modelling software, coded as a parallel application using the MPI framework.

Load balancing: To achieve good load balancing and minimize communication overhead, CHeart's topology decomposition strategy is based on an optimal graph partitioning strategy [43]. Briefly, the algorithm associates with each element a node. The nodes are linked by a set of edges if the associated elements belong to the same topology and are adjacent, or if the elements are related through an interface mapping, in the case of different topologies. The algorithm seeks to split the graph into N partitions, N being the number of processors, while minimizing the number of edge cuts. However, in order to account for variations in cost per element between topologies, as seen in multiphysics problems, each node also has an associated weight, based on the cost of assembling the element mass matrix.

In the case of a (S)PUFEM based system, the graph node set is generated based on the elements in $\mathcal{T}_b^h$, $\mathcal{T}_e^h$ and $\mathcal{T}_i^h = \mathcal{T}_{i,1}^h \cup \mathcal{T}_{i,2}^h$. The graph edge set is formed of intra-topology edges (generated based on the element adjacency in the $\mathcal{T}_b^h$ and $\mathcal{T}_e^h$) and inter-topology edges (where every element of $\mathcal{T}_i^h$ is linked with one background and one embedded parent). When considering the element cost associated with partitioning, we remark that $\mathcal{T}_i^h$ only acts as an interface and hence has no problem-driven requirement for a specific basis function setup. Hence, we generally associate with it a nominal linear basis function set. In the case of an (S)PUFEM problem, the resulting default ratio between the weighting associated with the intersection sub-elements and the standard background/embedded elements is thus given by the ratio between the number of quadrature nodes employed by each type. Note however, that this cost model does not account for differences related to: (1) a more costly evaluation of solution fields in the case of SPUFEM, and (2) the ability of the standard SFEM implementation to practically avoid the need of numerical integration. This issue will be further investigated in the Numerical Results section, where an additional cost scaling parameter is introduced to provide a tuning option.

Interface mapping cache: To enable the transfer of information from one grid to another, the original CHeart implementation permanently stored only a minimum amount of information, limited only to the mapping of nested element's nodes onto the master element space of the host element. In practice, this meant that each assembly loop based on a nested topology (i.e. $\mathcal{T}_i^h$) would require an *on the fly* mapping of the basis functions and their derivatives for all host topologies involved in the respective problem (i.e. $\mathcal{T}_b^h$ and $\mathcal{T}_e^h$). Hence, in order to speed up (S)PUFEM sub-problem assemblies, we implemented a new interface cache list type, which allows for the option to store these mappings for quick access.

3.5. Solver strategy

In CHeart, the non-linear systems represented by Eq. 7 and 16 are solved using the Shamanskii-Newton-Raphson method [44, 37]. This approach allows for the reuse of the Jacobian matrix and its inverse for multiple fixed-point iterations, as long as the norm of the residual decreases by a desired amount, resulting in a decrease of the computational cost. Furthermore, the matrix inverse and solution updates are computed using the direct solver MUMPS [45].

4. Numerical Results

The numerical results presented in this section focus on two primary goals: (1) evaluating the solution accuracy of new SPUFEM flow solver, comparing it with the more standard SFEM approach, and (2) the assessment of its efficiency in terms of run-time for different implementation and problem setup options defined in Section 3. In particular, we present three transient flow test cases: two examples in 2D, one with viscous dominated regime and the other for moderately high Reynolds numbers, and a viscous dominated 3D benchmark problem. (Furthermore, in *Appendix* Appendix A, these results are complemented with an analysis of the impact of the embedded mesh size in the case of a 2D benchmark problem.) These tests where carried on two clusters: ORCA for the 2D simulations and TOM for 3D, see Tab. 1.

4.1. 2D Unsteady Turek test

To compare the accuracy of both SPUFEM and boundary fitted SFEM implementations, we first considered the classic Schäfer and Turek benchmark [46], specifically, the 2D-2 transient test for $Re = 100$. The numerical experiment was run on 5 levels of refinement, where an increase in level is equivalent to the halving of the average element size. For each level, we constructed three grids (i.e. boundary fitted, background and embedded) of comparable quasi-homogeneous resolution in order to allow for an easier comparison. See Fig. 5 for a visual representation of the grid setup and Tab. 2 for a break down of the mesh statistics and number of cores applied in each level. The simulations were run for a total of 10 in-simulation non-dimensional time units, split into 1000 time steps of 0.01 time unit duration. The average inflow velocity was increased linearly in the first 20 time steps, after which it was kept fixed in time. For the integration over the interface element, we used as a sixth order accurate Lyness triangle quadrature scheme. Furthermore, the interface cache storing was not applied.

Four parameters were computed to help quantify solutions accuracy: the coefficients of drag (c_D) and lift (c_L), the pressure drop across the cylinder (Δp) and the Strouhal number (St). The estimations are based on the following parameter definitions:

$$\begin{aligned} c_D = \frac{2F_x}{\rho \bar{v}^2 A}, \quad c_L = \frac{2F_y}{\rho \bar{v}^2 A}, \quad St = \frac{fA}{\bar{v}}, \\ \Delta p = p(0.15, 0.2) - p(0.25, 0.2), \end{aligned} \tag{25}$$

where F_x and F_y are the net forces drag and lift acting on the obstacle, $\bar{v}$ is the average inflow velocity, A is the obstacle's cross sectional area perpendicular to the main flow direction (here A coincides with the diameter) and f is the wake shedding frequency (here computed as the inverse of the c_L oscillation period). The compilation of estimated values of c_D, c_L and Δp can be found in Tables 3 and 4. The parameter estimations are generally comparable, particularly in the case of higher resolutions where the absolute error between approaches is smaller than 1%. Furthermore, these values appear to converge to the literature values [46, 47], although the rate appears to be arguably slower for c_L. For level 2 through 5, both methods estimated the Strouhal number at 0.2941, slightly below the literature interval of 0.2950 to 0.3050, but acceptable given the relative coarseness of temporal discretization. Conversely, in the case of level 1 mesh refinement, neither approach displayed vortex shedding. A good agreement between the two methods can also be observed qualitatively, see Fig. 6. The SPUFEM solution appears to be smooth, with no artifacts in the coupling region, and the main characteristics of the flow, such as the recirculation area and the wake, display similar features for comparable grid resolutions.

The total simulation run times for the two approaches are compiled in Table 4. The results indicate that while the SPUFEM approach can be relatively expensive to run for coarse mesh levels, showing an approx. 97% higher run time for the coarsest level, the discrepancies between the two diminishes with refinement. The most probable explanation for this phenomenon is that as the refinement level increases the proportion of interface elements decreases, see Tab. 2, thus making the relative cost of the SPUFEM sub-problem smaller.

4.2. 2D high Reynolds number flow around obstacle test

To extend the comparison between the SFEM and SPUFEM approaches for more practical flow regimes, we adapt the benchmark problem presented in Section 4.1 as follows. The lateral wall boundary conditions are changed from no-slip to a zero normal component only. At the inflow, we impose a constant profile which increases linearly in the first 0.2 seconds of in-simulation time from null to $[30; 0]$ m/s. From this point, the simulation continues for another 0.3 seconds with constant inflow conditions such that a periodic steady state wake shedding pattern is achieved. With density and viscosity parameters set to 1.0 kg/m^3 and 10^{-3} Pa $\cdot$ s, and an obstacle size of 0.1 m, we obtain an approximate $Re = 3000$ flow regime. To run the simulations, we consider the same grids as Section 4.1, but limit the refinement level to two and above due to the relative coarseness of the first level. Furthermore, a constant time step size of 10^{-4} seconds was employed in all simulations.

In Figure 7, we display the solution fields obtained using both SPUFEM and SFEM approaches, at time points enumerated in Table 5. Note, at this higher Reynolds number regime, we were able to capture more complex flow features, specifically the fact that the wake bends periodically towards one wall or the other. However, we weren't able to match this behaviour with any specific feature in the transient behaviour of either c_D and c_L, shown in Figure 8 for the time interval $[4.0,\ 5.0]$ seconds. For this reason, the frames in Figure 7 where chosen on the basis that at the corresponding in-simulation time point the measured c_L is a local maximum and that wake is bent towards the right wall. In both Figures 7 and 8, we can see that the relative comparability of the two methods (for similar grid resolution) shows no significant deterioration as a result of changing the flow regime. The phase shift in the latter, can largely be explained by the fact that the onset of vertex shedding is not identical for the two method, with some delay observed in the case of SPUFEM.

4.3. 3D Unsteady Turek test

In this section, we extended the accuracy comparison between the SFEM and SPUFEM methods to transient 3D flow. Here, the numerical experiment takes the form of the 3D-3Z benchmark problem developed by Schäfer and Turek [46], and shares a similar characteristics to the simulations in Section 4.1. The domain consists of a rectangular channel of dimensions 2.5 m $\times$ 0.41 m $\times$ 0.41 m, with a cylindrical end-to-end obstacle of diameter $D = 0.1$ m placed along the $(x, y) = (0.5\text{ m}, 0.2\text{ m})$ axis, near the inflow surface. The inflow boundary condition is described by the function:

$$v_{inflow}(y, z, t) = 16 V_{max} yz(H - y)(H - z) \sin(\pi t/8)/H^2, \tag{26}$$

where $H = 0.41$ m and $V_{max} = 2.25$ m/s. The flow is observed over the $I = [0, 8]$ seconds time interval, the equivalent of one pulse. Furthermore, the viscosity and density are set to 10^{-3} Pa $\cdot$ s and 1 kg/m^3, resulting in a Reynolds number varying between 0 and 100.

For the two methods, we consider two equivalent levels of spatio-temporal discretization, see Tab. 6 for the statistics and number of cores used and Fig. 5 for an illustration of the mesh setup. To try to minimize the effect of geometric errors on the computation of the coefficient of lift, the grids were built such that they are quasi-symmetric with respect to the $z = 0.2$ plane, where the top half is slightly stretched above the obstacle to reach the appropriate height. In the case of SPUFEM, the intersection grids were

generated using the Sutherland-Hodgman approach. The numerical integrations are performed using the Keast quadrature scheme for integrations accurate for polynomials up to order 6.

In Fig. 9, we show that the numerical solutions obtained using the two methods are comparable for both velocity and pressure. In order to obtain a qualitative assessment, we again computed the coefficients of drag and lift as defined in Eq. 25, adjusting the cross sectional area to $A = DH$. In Figure 10, we plot the evolution in time of the two parameters. In the case of the coefficient of drag, the discrepancy between the methods are relatively small, with a relative difference of peak c_D estimations of 1.8% for Level 1 and 0.8% for Level 2. With respect to the benchmark values, the relative errors of the SPUFEM estimates were 14.7% and 2.5%. In the case of the coefficient of lift, the inter-method errors and the errors with respect to the benchmark are generally higher than 10%. We believe that the main reason for this is the fact that $c_D >> c_L$, meaning that the refinement errors are more likely to impact accuracy. However, despite these errors, some of the features of the c_L curve are retained in all simulations.

4.4. 3D Unsteady Turek: Efficiency and problem setup tests

In this section we examine the impact on the code efficiency of the different implementation and problem options described in Section 3. Briefly, these are:

1. the use/absence of an interface mapping cache, denoted as **U.C.** and **N.C.**, respectively
2. the interface triangulation algorithm, i.e. either Eberley's (**E.**) or Sutherland-Hodgman approaches (**S.H.**)
3. the lower order quadrature scheme, and
4. the partition weighting of the PUFEM sub-problem elements.

Here, we focus on the impact on the **residual assembly time** and **MPI waiting time**, two important metrics of solver efficiency, and also the primary targets of the options listed above. For reference, we also report the total run time, which includes the time used by the direct solver. However, the efficiency of the matrix solver is only partly influenced by the element weighting and not at all by the other options. Thus, reducing its associated cost is a separate question that does not fall under the scope of this study.

The options are tested in two stages, with the *first stage* focusing on options 1. (**N.C.** or **U.C.**) and 2. (**E.** or **S.H.**), using a quadrature scheme accurate for the integration of sixth order polynomials and a default intersection element weighting of one. Table 7 compiles the results for both 3D refinement levels using three different option permutations, namely: **N.C.** + **E.**, **U.C.** + **E.** and **U.C.** + **S.H.**, with the SFEM results added for reference. Comparing the **N.C.** + **E.** and **U.C.** + **E.**, we see significant benefits from using the cache, with marked time drops observed across almost all five metrics. These improvements are particularly noticeable in the case of the first level of refinement, where the total and average residual assembly times are almost halved, and the maximum MPI wait times and MPI wait time ratio are cut to almost a quarter of the original value. Here, the MPI wait time ratio denotes the ratio between the maximum and minimum times reported by the cores. The slight discrepancy between refinement levels, may be partially explained by the originally higher MPI ratio reported by the first level in the **N.C.** + **E.** case. Assuming the **U.C.** is always beneficial, we compare **U.C.** + **E.**

and **U.C.** + **S.H.** Looking at the direct outputs of the approaches, i.e. the intersection grids, we notice a significant decrease in the number of elements: from 1.12M to 0.44M for the first level, and from 3.12M to 1.20M. Thus, while in the case of **E.** the interface grid represents approximately three quarters of the total number of elements, the ratio drops to almost a half for **S.H.**. In practice, this translates to a further reduction of the residual and MPI time by 2 to 4 times, in Tab. 7. In general, we see that these options have a significantly lower impact on the total run time observed for level 2 refinement. The principal explanation for this is that as the system size decreases, the higher the running cost MUMPS becomes as proportion of the total. Hence any impact of the two options is lower.

In the *second stage* of testing we compare the impact of different partition weights and integration schemes, with the results compiled in Table 8. Here the 3D Turek benchmark is run on the coarser grid set in 16 different configurations, resulting from the permutation of 4 interface element weighting factors and 4 numerical integration schemes. The cache and **S.H.** options are also employed in all cases to reduce cost. The use of lower order quadrature schemes is motivated by the fact that they introduce negligible errors, with the relative L^2 errors over time ranging between 10^{-9} and 10^{-12}. Fixing the weighting at 1, we see that changing the quadrature can result in lower residual times, from 5.7k at order 6 to 2.8k at order 2. Significantly, the MPI wait time appears to vary non-monotonously while the MPI ratio generally increases when lowering the quadrature order. Thus, lowering the quadrature on its own is only partially beneficial as it can result in poorer load balancing. Conversely, by increasing the partition weighting of the SPUFEM sub-problem elements, we see a significant improvement of overall efficiency and load balancing with: (1) a reduction of the residual assembly times by almost a half and (2) bringing the MPI maximum wait time and ratio values to orders similar to those reported in the SFEM case. Note however, that the optimum combination of parameters varies across the 5 time metrics. This indicates that in order to consistently improve problem run times, we will need to have a better understanding of how SPUFEM affects the communication time and the linear algebra solver.

5. Conclusion

A new stabilized SPUFEM solver for moderate and high Reynolds number flows was presented, combining the partition of unity framework for domain decomposition with the cG(1)cG(1) stabilized scheme. Furthermore, we outlined a practical implementation extension of the method to 3D, including a number of avenues to improve the process of residual assembly: the use of a memory cache, the splitting strategy, the lower quadrature schemes and the tuning of the partition weighting for interface elements.

SPUFEM's effectiveness is shown to be comparable to that of SFEM in the case of three example problems, including both 2D and 3D standard Schaefer-Turek benchmarks, as well as a $Re = 3000$ adapted version of the same benchmark. These similarities are shown both in qualitative terms, with certain flow features appearing in the flow solutions for approximately matching grid resolution, as well as quantitatively. Thus, for the 2D and 3D problems, the estimated system parameters, such as the coefficient of drag, are shown to converge to the values available from literature. Similarly, in the case of the $Re = 3000$, SPUFEM is shown to closely follow the estimations of c_d and c_f offered

by SFEM, both in terms of range and average values, as well as in terms of transient behaviour.

Adapting the 2D Turek benchmark, SPUFEM is shown in Appendix A to be robust to changes in the width of the embedded grid. In particular, the results indicate that reducing the thickness of the grid to three or four element layers has a very limited impact on the quality and accuracy of the solution, while decreasing the number of total elements involved in the computations, particularly in the intersection grid. If this result is proven to hold for more general 3D cases and for different ratio of element size between the background and embedded grid, this practice may simplify the mesh generation process due to the increased flexibility and may also prove an effective means of reducing the cost of the problem.

Using the 3D benchmark problem as study case, the four implementation options are shown to have a significant impact on improving the solver's efficiency both in terms of reducing the residual assembly time as well as as improving load balancing across cores. These options are shown to be beneficial on their own, with the cache and lower quadrature schemes reducing the cost per element, the Sutherland-Hodgman algorithm reducing the number of elements, and the weighting parameter being used to account for discrepancies not captured by the standard CHeart average cost model. However, more significantly, the combination of these options are shown to have a synergistic effect, lowering the residual assembly time by an order of magnitude and improving the load balancing to levels comparable to the more standard SFEM approach. While the runtime of SPUFEM is still about 3 times larger than that of the classical stabilized approach, this case is not necessarily illustrative of some of the advantages of our method. Primarily, we believe that this cost may be offset by the greater flexibility of SPUFEM in the context of large deformation FSI problems, in which the use of standard boundary fitted approaches is either challenging or impractical. Secondly, the additional cost of the SPUFEM problem is application specific and it depends on both the relative surface area of the embedded object and on the volume of the embedded domain. The latter, as seen in Appendix A, may be significantly reduced without impacting the method's accuracy.

To further improve the cost effectiveness of (S)PUFEM, future work will need to focus on ways to lower the cost of inverting the Jacobian matrix, as highlighted by the 3D simulation with level 2 refinement. Potential avenues to address this include adapting (S)PUFEM into: (1) pressure segregated schemes, accompanied by a switch from direct to preconditioned iterative solvers, and (2) semi-implicit/explicit coupling schemes. Another avenue for extending this work will be to study the robustness of the method for different ratios of background and embedded element sizes and how this may be impacted by the choice of ψ, particularly looking into increasing the width of the transition band and/or increasing its polynomial order. Moving to more complex moving domains and FSI applications, the current implementation will also need to be able to handle time dependent overlap configurations and to be able to generate intersection grids on the fly, a scenario where one can expect that the (S)PUFEM specific operations that we sought to optimize in this paper will be a more dominant part of the computational cost.

Acknowledgements

D.N. acknowledges funding from the Engineering and Physical Sciences Research Council, United Kingdom (EP/N011554/1 and EP/R003866/1). A.M. gratefully acknowledges financial support from the Swedish Research Council under Starting Grant 2017-05038 and from the Wenner-Gren foundation, Sweden under travel grant SSh2017-0013. JH acknowledges the financial support of the Swedish Research Council under Grant 2018-04854. This work is funded by the King's College London and Imperial College London EPSRC Centre for Doctoral Training in Medical Imaging (EP/L015226/1). This work is supported by the Wellcome EPSRC Centre for Medical Engineering at King's College London (WT 203148/Z/16/Z) and by the National Institute for Health Research (NIHR) Biomedical Research Centre award to Guy and St Thomas' NHS Foundation Trust in partnership with King's College London. The views expressed are those of the authors and not necessarily those of the NHS, the NIHR or the Department of Health.

References

[1] C. S. Peskin, Flow patterns around heart valves: a numerical method, Journal of computational physics 10 (2) (1972) 252–271.

[2] L. Zhang, A. Gerstenberger, X. Wang, W. K. Liu, Immersed finite element method, Computer Methods in Applied Mechanics and Engineering 193 (21-22) (2004) 2051–2067.

[3] L. Zhang, M. Gay, Immersed finite element method for fluid-structure interactions, J. Fluids Struct. 23 (6) (2007) 839–857.

[4] D. Boffi, L. Gastaldi, A finite element approach for the immersed boundary method, Comput. Struct. 81 (8) (2003) 491–501.

[5] R. Glowinski, T.-W. Pan, J. Periaux, A fictitious domain method for Dirichlet problem and applications, Computer Methods in Applied Mechanics and Engineering 111 (3-4) (1994) 283–303.

[6] R. Glowinski, T.-W. Pan, T. I. Hesla, D. D. Joseph, A distributed Lagrange multiplier/fictitious domain method for particulate flows, International Journal of Multiphase Flow 25 (5) (1999) 755–794.

[7] W. A. Wall, P. Gamnitzer, A. Gerstenberger, Fluid–structure interaction approaches on fixed grids based on two different domain decomposition ideas, International Journal of Computational Fluid Dynamics 22 (6) (2008) 411–427.

[8] R. Van Loon, P. D. Anderson, J. De Hart, F. P. Baaijens, A combined fictitious domain/adaptive meshing method for fluid–structure interaction in heart valves, International Journal for Numerical Methods in Fluids 46 (5) (2004) 533–544.

[9] D. Kamensky, M.-C. Hsu, Y. Yu, J. A. Evans, M. S. Sacks, T. J. Hughes, Immersogeometric cardiovascular fluid–structure interaction analysis with divergence-conforming B-splines, Computer methods in applied mechanics and engineering 314 (2017) 408–472.

[10] A. Gerstenberger, W. A. Wall, An extended finite element method/Lagrange multiplier based approach for fluid–structure interaction, Computer Methods in Applied Mechanics and Engineering 197 (19-20) (2008) 1699–1714.

[11] A. Massing, M. G. Larson, A. Logg, Efficient implementation of finite element methods on nonmatching and overlapping meshes in three dimensions, SIAM Journal on Scientific Computing 35 (1) (2013) C23–C47.

[12] F. Alauzet, B. Fabrèges, M. A. Fernández, M. Landajuela, Nitsche-XFEM for the coupling of an incompressible fluid with immersed thin-walled structures, Computer Methods in Applied Mechanics and Engineering 301 (2016) 300–335.

[13] B. Schott, Stabilized cut finite element methods for complex interface coupled flow problems, Ph.D. thesis, Technical University of Munich (2017).

[14] A. Massing, B. Schott, W. Wall, A stabilized Nitsche cut finite element method for the Oseen problem, Comput. Methods Appl. Mech. Engrg. 328 (2018) 262–300.

[15] E. Burman, S. Frei, A. Massing, Eulerian time-stepping schemes for the non-stationary stokes equations on time-dependent domains, arXiv preprint arXiv:1910.03054 (2019).

[16] A. Verkaik, M. Hulsen, A. Bogaerds, F. van de Vosse, An overlapping domain technique coupling spectral and finite elements for fluid flow, Computers & Fluids 100 (2014) 336–346.
[17] A. Massing, M. G. Larson, A. Logg, M. Rognes, A Nitsche-based cut finite element method for a fluid-structure interaction problem, Commun. Appl. Math. Comput. Sci. 10 (2) (2015) 97–120.
[18] S. Shahmiri, A. Gerstenberger, W. A. Wall, An xfem-based embedding mesh technique for incompressible viscous flows, International Journal for Numerical Methods in Fluids 65 (1-3) (2011) 166–190.
[19] B. Schott, C. Ager, W. A. Wall, Monolithic cut finite element–based approaches for fluid-structure interaction, International Journal for Numerical Methods in Engineering 119 (8) (2019) 757–796.
[20] J. Benek, J. Steger, F. C. Dougherty, A flexible grid embedding technique with application to the Euler equations, in: 6th Computational Fluid Dynamics Conference Danvers, 1983, p. 1944.
[21] J. Steger, F. Dougherty, J. Benek, A Chimera grid scheme, in: Advances in Grid Generation, Vol. ASME FED-5, 1983, pp. 59–69.
[22] J. L. Steger, J. A. Benek, On the use of composite grid schemes in computational aerodynamics, Computer Methods in Applied Mechanics and Engineering 64 (1-3) (1987) 301–320.
[23] M. Balmus, A. Massing, J. Hoffman, R. Razavi, D. A. Nordsletten, A partition of unity approach to fluid mechanics and fluid–structure interaction, Computer Methods in Applied Mechanics and Engineering 362 (2020) 112842.
[24] G. Karniadakis, S. Sherwin, Spectral/hp element methods for computational fluid dynamics, Oxford University Press, 2013.
[25] H. Xu, C. D. Cantwell, C. Monteserin, C. Eskilsson, A. P. Engsig-Karup, S. J. Sherwin, Spectral/hp element methods: Recent developments, applications, and perspectives, Journal of Hydrodynamics 30 (1) (2018) 1–22.
[26] A. N. Brooks, T. J. Hughes, Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations, Computer methods in applied mechanics and engineering 32 (1-3) (1982) 199–259.
[27] T. J. Hughes, T. Tezduyar, Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible euler equations, Computer methods in applied mechanics and engineering 45 (1-3) (1984) 217–284.
[28] J. Hoffman, C. Johnson, A new approach to computational turbulence modeling, Computer Methods in Applied Mechanics and Engineering 195 (23-24) (2006) 2865–2880.
[29] Y. Bazilevs, V. Calo, J. Cottrell, T. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Computer methods in applied mechanics and engineering 197 (1-4) (2007) 173–201.
[30] Y. Bazilevs, I. Akkerman, Large eddy simulation of turbulent taylor–couette flow using isogeometric analysis and the residual-based variational multiscale method, Journal of Computational Physics 229 (9) (2010) 3402–3414.
[31] T. J. Hughes, L. Mazzei, K. E. Jansen, Large eddy simulation and the variational multiscale method, Computing and visualization in science 3 (1) (2000) 47–59.
[32] F. Brezzi, J. Pitkäranta, On the stabilization of finite element approximations of the stokes equations, in: Efficient solutions of elliptic systems, Springer, 1984, pp. 11–19.
[33] E. Burman, M. Fernández, P. Hansbo, Continuous interior penalty finite element method for Oseen's equations, SIAM J. Numer. Anal. 44 (3) (2006) 1248–1274.
[34] R. Codina, J. Blasco, Stabilized finite element method for the transient navier–stokes equations based on a pressure gradient projection, Computer Methods in Applied Mechanics and Engineering 182 (3-4) (2000) 277–300.
[35] P. Farrell, J. Maddison, Conservative interpolation between volume meshes by local Galerkin projection, Computer Methods in Applied Mechanics and Engineering 200 (1-4) (2011) 89–100.
[36] F. Brezzi, R. S. Falk, Stability of higher-order Hood–Taylor methods, SIAM Journal on Numerical Analysis 28 (3) (1991) 581–590.
[37] A. Hessenthaler, O. Röhrle, D. Nordsletten, Validation of a non-conforming monolithic fluid-structure interaction method using phase-contrast MRI, International journal for numerical methods in biomedical engineering 33 (8) (2017) e2845.
[38] T. E. Tezduyar, Y. Osawa, Finite element stabilization parameters computed from element matrices and vectors, Computer Methods in Applied Mechanics and Engineering 190 (3-4) (2000) 411–430.
[39] M. Braack, E. Burman, V. John, G. Lube, Stabilized finite element methods for the generalized Oseen problem, Comput. Methods Appl. Mech. Engrg. 196 (4) (2007) 853–866.
[40] U. M. Mayer, A. Gerstenberger, W. A. Wall, Interface handling for three-dimensional higher-order XFEM-computations in fluid–structure interaction, International Journal for Numerical Methods

in Engineering 79 (7) (2009) 846–869.
[41] Y. Sudhakar, W. A. Wall, Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods, Computer Methods in Applied Mechanics and Engineering 258 (2013) 39–54.
[42] J. Lee, A. Cookson, I. Roy, E. Kerfoot, L. Asner, G. Vigueras, T. Sochi, S. Deparis, C. Michler, N. P. Smith, et al., Multiphysics computational modeling in CHeart, SIAM Journal on Scientific Computing 38 (3) (2016) C150–C178.
[43] G. Karypis, V. Kumar, A parallel algorithm for multilevel graph partitioning and sparse matrix ordering, Journal of Parallel and Distributed Computing 48 (1) (1998) 71–95.
[44] V. Shamanskii, A modification of Newton's method, Ukrainian Mathematical Journal 19 (1) (1967) 118–122.
[45] P. R. Amestoy, I. S. Duff, J.-Y. L'excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers, Computer methods in applied mechanics and engineering 184 (2-4) (2000) 501–520.
[46] Schäfer, Michael and Turek, Stefan and Durst, Franz and Krause, Egon and Rannacher, Rolf, Benchmark computations of laminar flow around a cylinder, in: Flow simulation with high-performance computers II, Springer, 1996, pp. 547–566.
[47] FEATFLOW Finite Element Software for the Incompressible Navier-Stokes Equations.
URL www.featflow.de

Name	Orca	TOM
OS	Ubuntu 16.04	SUSE SLES 11 SP1
CPUs	1x AMD Ryzen Threadripper 2990WX 3.0 GHz, 32 cores	76x Intel(R) Xeon(R) E7-8837, 2.66 GHz, 8 cores, Westmere EX
Memory per Node	128 GB	128 GB
Network	-	NUMAlink 5 Interconnect
Tot. processors	32	608

Table 1: The hardware and software specifications of the ORCA and TOM clusters.

	Level 1	Level 2	Level 3	Level 4	Level 5
SFEM					
No. elem	5348	21394	85368	342362	1369660
DOF	8445	32937	129744	516924	2061255
h_{min}	1.7E-2	7.9E-3	3.7E-3	1.9E-3	9.3E-4
Background					
No. elem	5390	21560	86212	346044	1384168
DOF	8487	33141	130917	522258	2082636
h_{min}	1.9E-2	9.2E-3	4.4E-3	1.9E-3	9.2E-4
Embedded					
No. elem	400	1608	6144	24140	94566
DOF	696	2604	9600	36978	143385
h_{min}	1.7E-2	8.5E-3	3.8E-3	1.9E-3	9.0E-4
Intersection					
No. elem	870	1771	3577	7203	14349
No. cores	4	4	8	8	16

Table 2: Grid statistics and number of cores used in the 2D Turek benchmark test for the SFEM and SPUFEM (for background, embedded, and intersection grids).

Level	max c_D		min c_D		max c_L		min c_L	
	SFEM	**SPUFEM**	**SFEM**	**SPUFEM**	**SFEM**	**SPUFEM**	**SFEM**	**SPUFEM**
1	2.73360	2.83450	2.73360	2.82783	-0.03546	-0.01249	-0.03547	-0.01249
2	2.87240	2.86849	2.86928	2.86692	0.20725	0.18443	-0.23827	-0.21275
3	3.07853	3.07804	3.05751	3.05933	0.54898	0.53006	-0.57965	-0.56186
4	3.17065	3.16951	3.12973	3.12871	0.77593	0.77343	-0.81010	-0.80885
5	3.20663	3.20634	3.15424	3.15384	0.88760	0.89058	-0.92551	-0.92410
Ref. [46]	3.22-3.24		N.A.		0.99-1.01		N.A.	
Ref. [47]	3.2271		3.1643		0.98658		-1.0213	

Table 3: Maximum and minimum values of the coefficient of drag and lift for the 2D-2 benchmark estimated using the SFEM and SPUFEM approaches.

Level	Δp		Run time (seconds)		Ratio (%)
	SFEM	**SPUFEM**	**SFEM**	**SPUFEM**	
1	1.85557	1.86424	43.579	85.912	97.1
2	2.12844	2.12644	459.48	580.20	26.3
3	2.31546	2.31218	1333.8	1843.0	38.2
4	2.40304	2.40292	7671.6	9845.2	28.3
5	2.44578	2.44556	24862	28810	15.9
Ref. [46]	2.46 - 2.50				

Table 4: 2D-2 benchmark pressure drop estimations, total run times and relative additional run time used in SPUFEM compared to SFEM.

Level	**SFEM** (sec.)	**SPUFEM** (sec.)
2	4.798	4.887
3	4.865	4.889
4	4.327	4.039
5	4.813	4.917

Table 5: The time points corresponding to the frames in Figure 7. These were chosen based on the fact that the value of c_L is a local maximum and (where applicable) the wake is bent towards the right wall.

	Level	Processors	Mesh type	No. elems.	DOF.	h_{min}	Δt
SFEM	1	32		437978	314960	0.0114	1/100
	2	124		1153998	807644	0.0061	1/200
SPUFEM	1	32	Background	412978	296708	0.0119	
			Embedded.	28406	22716	0.0137	1/100
			Intersection	443521	-	-	
	2	128	Background	1223462	861892	0.0059	
			Embedded.	67080	53668	0.0056	1/200
			Intersection	1199310	-	-	

Table 6: Mesh statistics, time steps sizes, and the number of processors used in the 3D Turek benchmark. Here, the SPUFEM interface grid was generated using the Sutherland-Hodgman algorithm.

Level	**N.C.+E.**	**U.C.+E.**	**U.C.+S.H.**	**SFEM**	**N.C.+E.**	**U.C.+E.**	**U.C.+S.H.**	**SFEM**
	Total run time (1000 s)				**Total residual assembly time (1000 s)**			
1	38.9	19.1	10.9	3.18	22.7	13.2	5.70	0.321
2	69.1	58.4	40.2	20.2	27.9	17.3	4.49	0.361
	Average residual assembly time (s)				**Max. MPI wait time (1000 s)**			
1	5.50	2.05	0.865	0.071	19.7	5.28	4.18	0.143
2	3.61	2.37	0.613	0.057	30.8	23.8	10.6	1.36
	MPI wait time max. ratio							
1	104.8	26.5	25.2	2.7				
2	24.6	34.9	15.7	2.8				

Table 7: Run time statistics for the 3D Turek test computed using SPUFEM for three permutations of implementation options 1 and 2. SFEM run time statistics for similar resolutions included as reference.

SPUFEM								
Total run time (s)					**Total residual assembly time (s)**			
P.W. \ **O.Q.**	2	3	4	5	2	3	4	5
1	9118.2	8588.2	9816.3	9439.2	3792.5	3909.2	4836.9	4377.7
2	8398.0	9092.6	**8522.5**	9333.5	3286.7	3615.3	3951.0	3849.1
3	8057.9	**8832.5**	9182.3	**9258.3**	3117.7	**3320.4**	3875.7	**3655.1**
4	**7775.5**	9442.6	10252	11092	**2903.7**	3432.8	**3527.3**	3928.0
Average residual assembly time (ms)					**Max. MPI wait time (s)**			
1	585.4	638.0	737.7	710.6	3646.8	3063.1	3358.4	3051.6
2	521.9	575.0	647.7	637.6	2162.2	1662.3	1247.7	1086.7
3	476.7	**528.6**	624.9	**612.1**	1364.5	473.47	**242.30**	**254.24**
4	**466.6**	539.8	**587.4**	633.9	**684.93**	**342.01**	1367.0	282.36
MPI wait time max. ratio								
1	36.7	31.7	32.0	30.4				
2	21.1	16.5	9.2	9.5				
3	11.7	3.6	**1.6**	**1.6**				
4	**6.1**	**2.2**	8.8	1.6				

Table 8: Simulation run time statics as functions of PUFEM sub-element partition weighting (**P.W.**) and the order of the quadrature used in the numerical integration (**O.Q.**). These results are based on varying the two parameters in the case of the Level 1 refinement PUFEM simulation. The cells marked in **red** indicate the minimum value achieved for a given quadrature scheme.

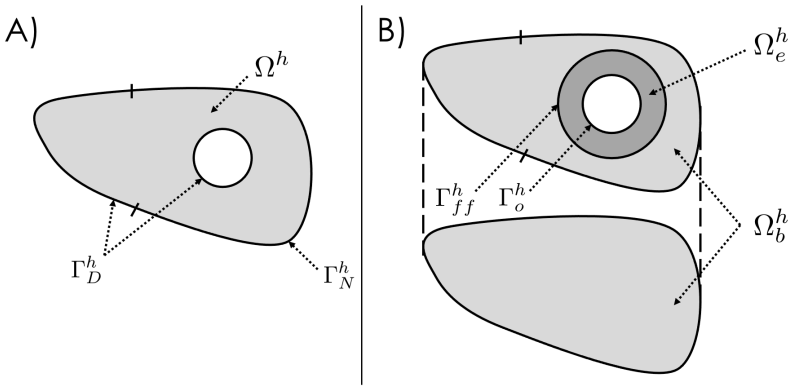

Figure 1: A) General representation of the domain for FEM setting with labels for the boundary regions. B) Analogous representation of the PUFEM setting for the same problem, with additional labels for the overlapping domains, obstacle boundary region and fluid-fluid boundary.

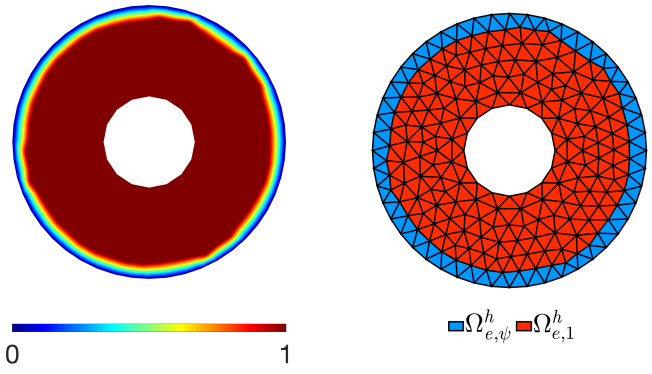

Figure 2: Example weighting field (left) and the resulting $\Omega_{e,\psi}^h$ and $\Omega_{e,1}^h$ subdomains.

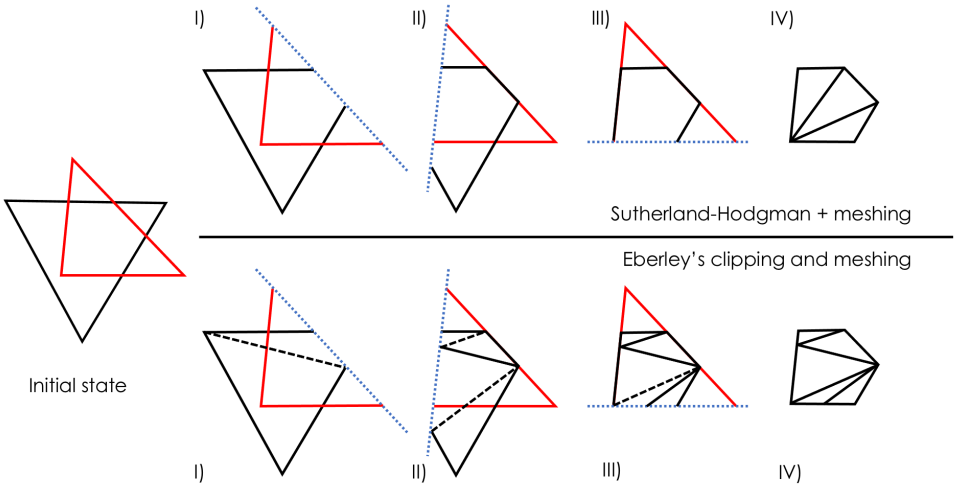

Figure 3: Illustrations of the Sutherlang-Hodgman and Eberley's approaches to the clipping and meshing of the overlap area between two triangles.

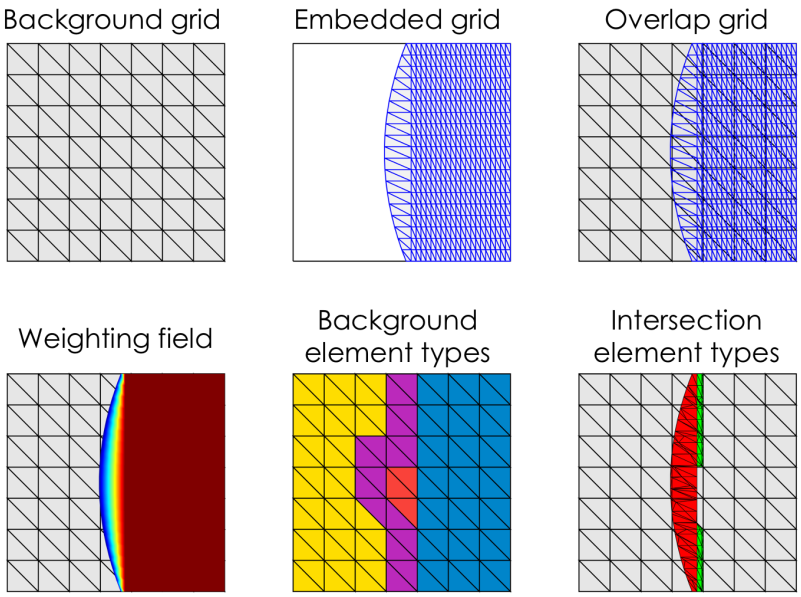

Figure 4: Illustration of sub-element generation and categorization for two grids. (Top) The background (left) and embedded (centre) grids marked in black and blue edges, respectively, as well as the overlap configuration. (Bottom-left) The weighting field in relation to the background grid. (Bottom-centre) Background element categorization based on their relation to the embedded grid: **yellow** for $\mathcal{T}^h_{b,b}$, **purple** for $\mathcal{T}^h_{b,c}$, **orange** for $\mathcal{T}^h_{b,d}$ and **blue** for $\mathcal{T}^h_{b,e}$. (Bottom-right) The resulting intersection grid overlaid on top of the background grid with color coding to distinguish the type: **red** for $\mathcal{T}^h_{i,\psi}$, i.e. elements used to both subtract unweighted contributions from cut background elements and PUFEM contributions, and **green** for $\mathcal{T}^h_{i,1}$, i.e. elements used in the subtraction process only.

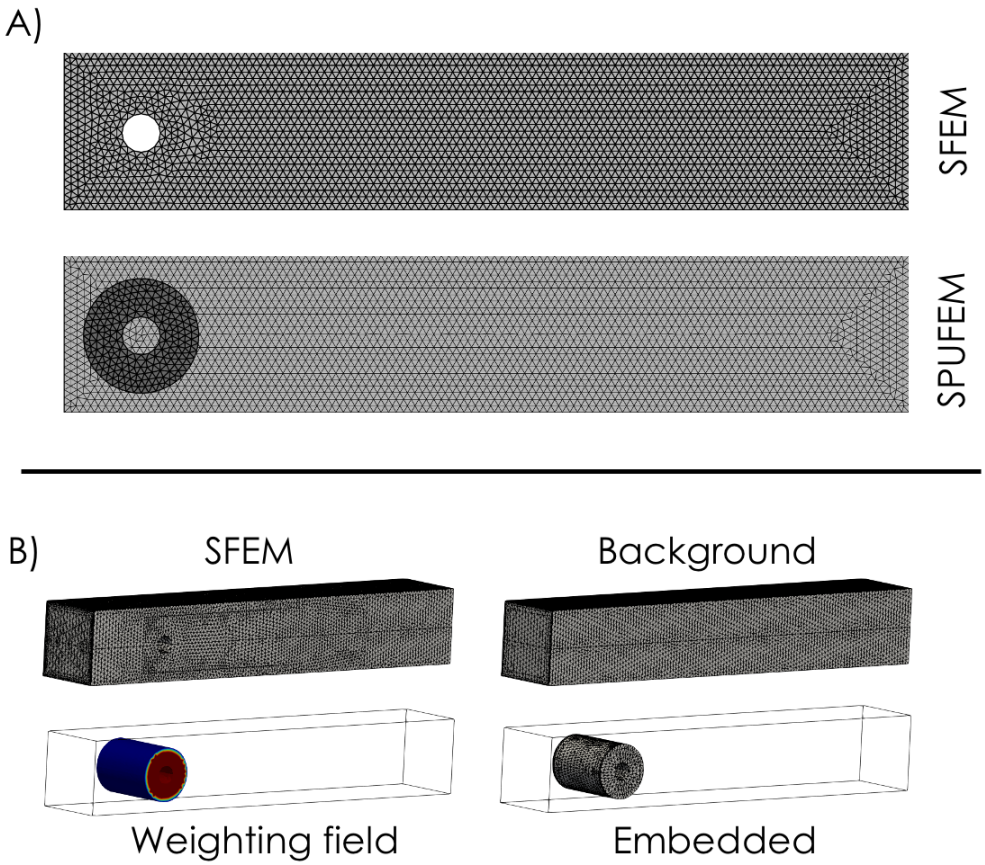

Figure 5: (Top) Example mesh setup used the in 2D Turek benchmark for both SFEM and SPUFEM approaches. (Bottom) Analogous setup for the 3D benchmark including the weighting field (ψ) as well.

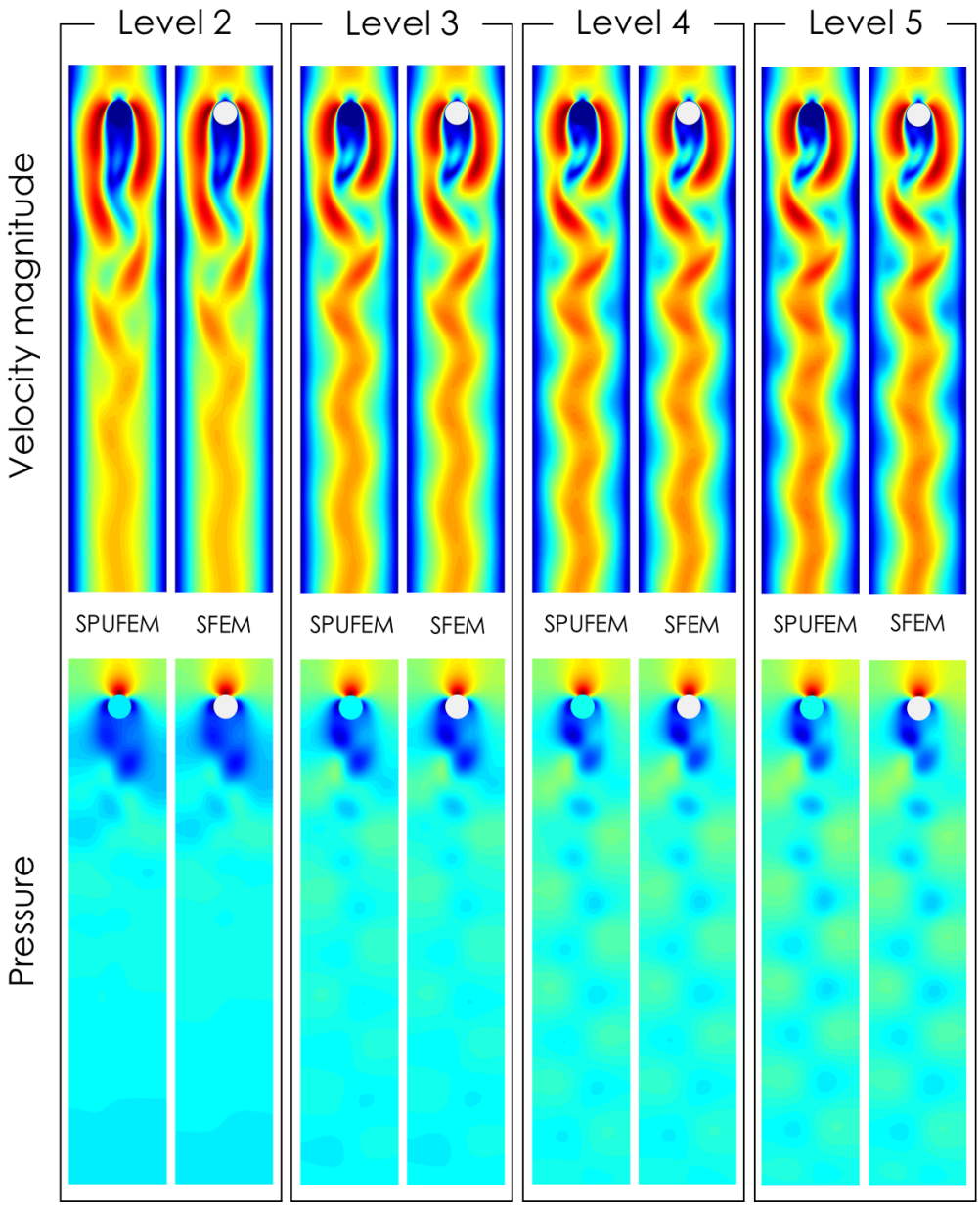

Figure 6: Fluid velocity magnitude and pressure fields captured at maximum positive c_L, using both SFEM and SPUFEM approaches, for four levels of refinement.

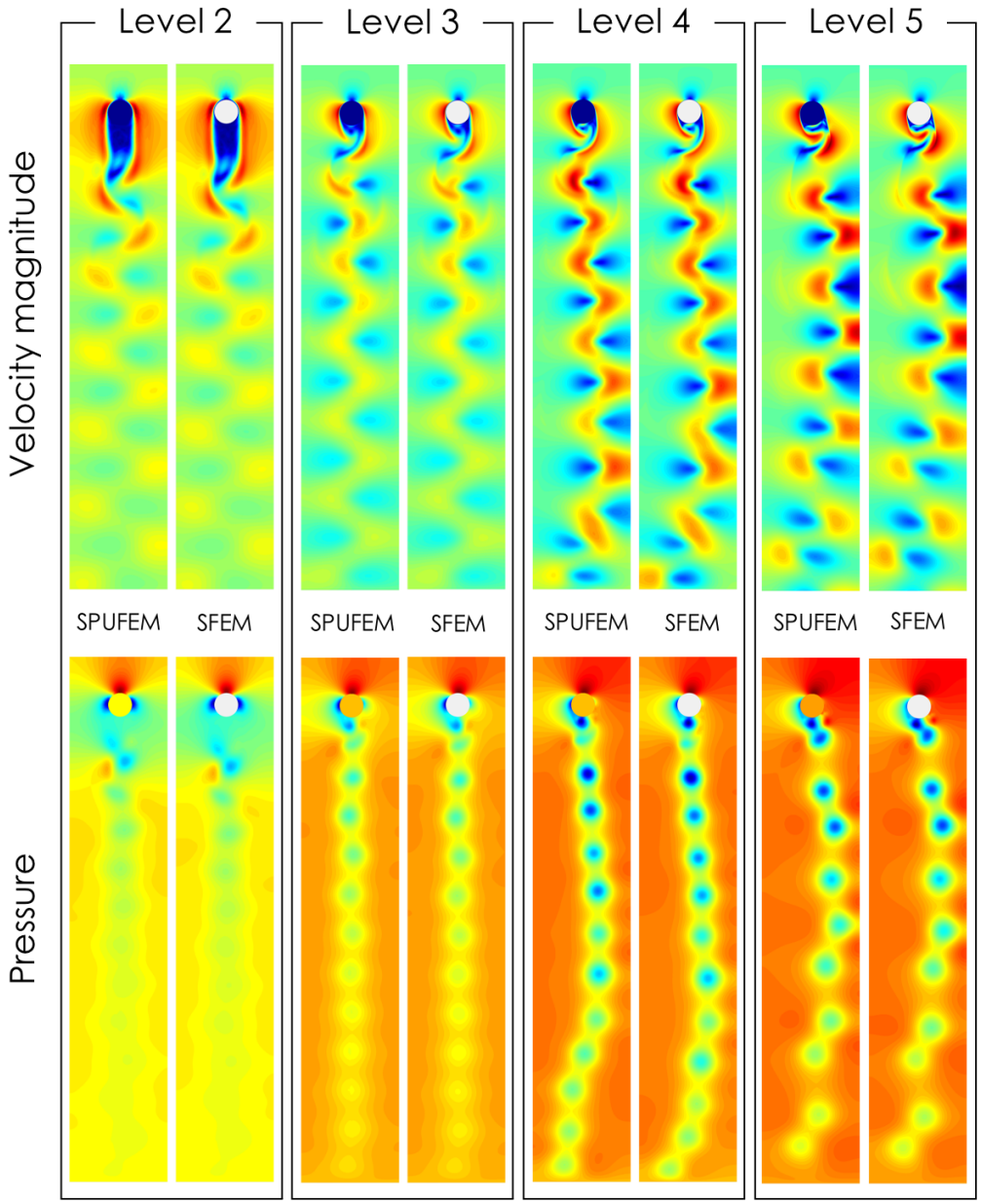

Figure 7: Fluid velocity magnitude and pressure fields captured at local maxim~~um~~a of c_L (see Tab. ...) for the $Re = 3000$ test. Solutions are computed with both SFEM and SPUFEM approaches, with four out five levels of refinement being represented here. In the case of Levels 4 and 5, the wake also oscillates from left to right.

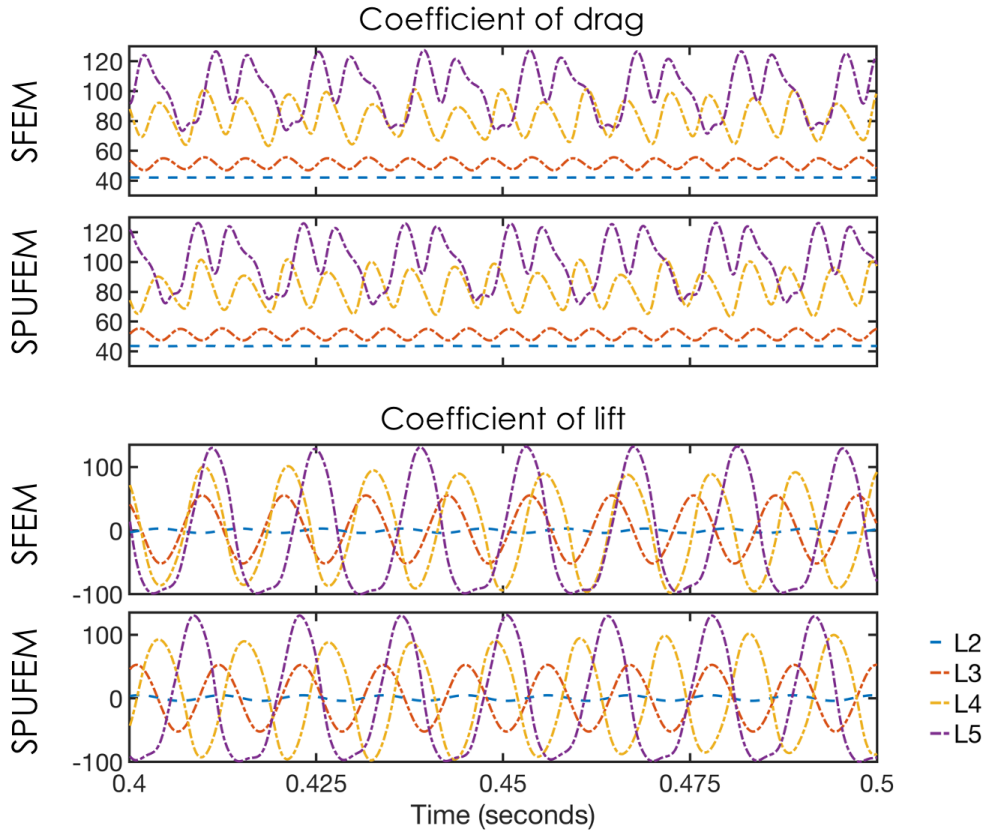

Figure 8: Estimated values for the coefficients of drag and lift in the last 0.1 seconds of the 2D high Reynolds number benchmark. Values were computed based on simulations using both boundary fitted SFEM and SPUFEM stabilized implementations. The legend is used to indicate the grid's resolution level, see 2.

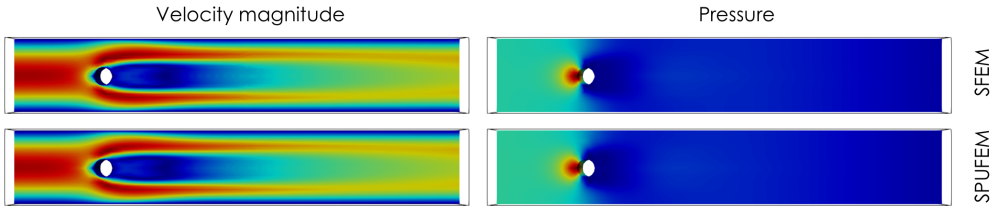

Figure 9: Solution fields for the 3D-3Z Turek benchmark computed with both boundary fitted and PUFEM approaches using Level 1 grids.

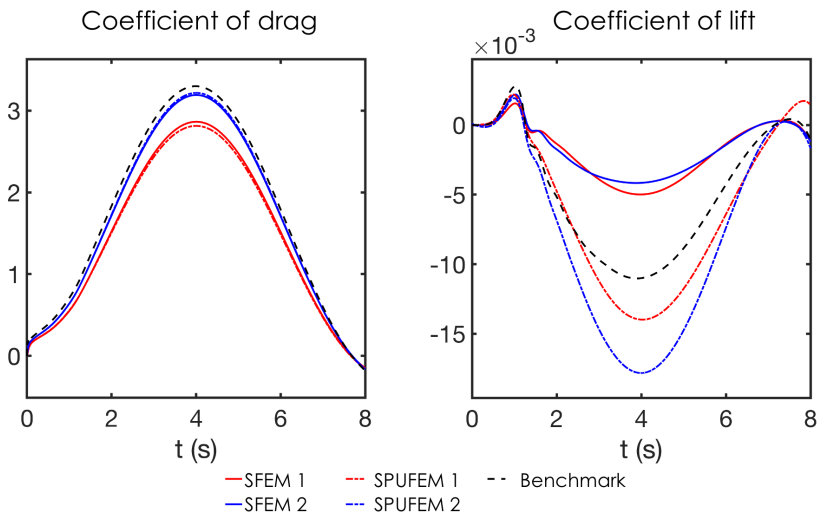

Figure 10: The coefficients of drag and lift over time for the 3D-3Z as computed with the boundary fitted and PUFEM methods. The reference literature values were obtained from [47].

Appendix:
A stabilized multidomain partition of unity approach to solving imcompressible viscous flow

Maximilian Balmus[a,*], Johan Hoffamn[b], André Massing[c], David Nordsletten[a,d]

[a] *School of Biomedical Engineering and Imaging Sciences, King's College London, 4th FL Rayne Institute, St Thomas Hospital, London, SE1 7EH, United Kingdom*
[b] *Division of Computational Science and Technology, KTH Royal Institute of Technology, SE-10044, Stockholm, Sweden*
[c] *Department of Mathematical Sciences, Norwegian University of Science and Technology, NO-74921, Trondheim, Norway* [d] *Department of Biomedical Engineering and Cardiac Surgery, University of Michigan, NCRC B20, 2800 Plymouth Rd, Ann Arbor, 48109*

Appendix A. Impact of restricted overlap

To examine the influence of the size of the overlap area on the accuracy of SPUFEM, we expand the Turek benchmark test by considering problems in which the embedded grid has a fixed number of elements over all levels of refinement. Specifically, we consider the cases where the embedded grid is formed of three and four elements across: 2 boundary layer elements (first being a quarter of the element size and the second a half) and one or two additional normal-shaped element layers, see Fig. A.1. As before the last of these layers, the one bordered by Γ^h_{ff} functions as transition band, i.e. Ω^h_ψ. Similarly to the previous section, where the test was performed on embedded regions of constant thickness, i.e. 0.15, simulations were run for five levels of refinement, once for each embedded grid type. The number of grid elements for each level, both for the embedded and resulting intersection grid, are compiled in Tab. A.1.

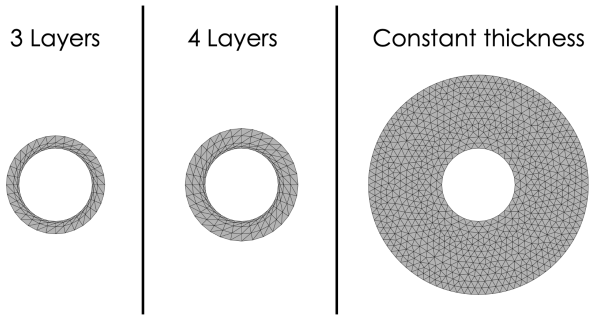

Figure A.1: Examples of two embedded meshes with restricted overlap area, i.e. three (left) and four (center) element layers, as well as a reference constant thickness mesh (right) used in Section 4.1.

An example of the resulting magnitude fields can be seen in Figure A.2, where the new results are compared with SFEM and SPUFEM (with constant embedded grid thickness) for reference. As it can be seen the fields are generally very similar, with the results almost indistinguishable for higher levels of refinement, suggesting that restricting the region of the embedded area has little influence on the quality of the results. This is also confirmed quantitatively, when comparing the estimations of the coefficients of drag and lift, see

Level	Embedded		Intersection	
	3 layers	4 layers	3 layers	4 layers
1	96	128	430	523
2	192	256	821	921
3	384	512	1735	1895
4	768	1024	4161	4399
5	1536	2048	11252	11771

Table A.1: Number of elements compiled for 3 and 4 layer embedded grids, as well as the resulting intersection grid. Values listed for refinement levels 1 to 5.

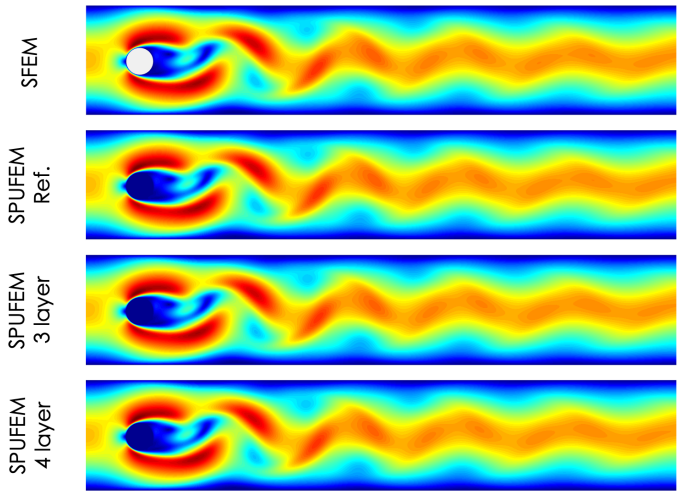

Figure A.2: The velocity magnitude in the Turek benchmark obtained using SFEM, SPUFEM with constant embedded are (i.e. Ref.), SPUFEM with embedded mesh of 3 layers, and 4 layers respectively. All four simulations are using the fifth refinement level mesh set.

Tab. A.2, with both cases producing comparable estimations to our previous SPUFEM estimations and also converge to the reference values, see Tab. 3.

Level	min c_D		max c_D		min c_L		max c_L	
	3 layers	4 layers	3 layers	4 layers	3 layers	4 layers	3 layers	4 layers
1	4.3417	3.2409	4.3453	3.2409	-0.4137	-0.2438	-0.3752	-0.2438
2	3.0330	3.0266	3.0344	3.0287	-0.4601	-0.3701	0.1426	-0.0173
3	3.0946	3.1612	3.1263	3.1925	-0.6961	-0.6963	0.5299	0.5698
4	3.1846	3.1729	3.2306	3.2180	-0.8540	-0.8580	0.8202	0.8222
5	3.1848	3.1919	3.2384	3.2485	-0.9473	-0.9445	0.9222	0.9319

Table A.2: Coefficients of drag and lift estimations using SPUFEM with embedded grid of 3 and 4 layers.